L Number	Hits	Search Text	DB	Time stamp
-	66950	web same form$1	USPAT	2003/10/01 13:11
-	3315	(web same form$1) and saving	USPAT	2003/09/30 14:50
-	872	((web same form$1) and saving) and storing	USPAT	2003/09/30 14:50
-	116	(((web same form$1) and saving) and storing) and caching	USPAT	2003/09/30 14:58
-	10247	www or "world wide web"	USPAT	2003/09/30 14:59
-	236	(www or "world wide web") and (fill-out or fill adj out) and form$1	USPAT	2003/09/30 15:01
-	12	((www or "world wide web") and (fill-out or fill adj out) and form$1) and ("fill-in" or fill adj in)	USPAT	2003/09/30 15:01
-	1	("5640577").PN.	USPAT	2003/10/01 10:23
-	145	form$1 adj completion	USPAT	2003/10/01 10:28
-	212	(715/507).CCLS.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/10/01 10:28
-	121	(715/505).CCLS.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/10/01 10:28
-	212	(715/507).CCLS.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/10/01 10:28
-	1514	clipboard or (clip adj board) or "clip-board"	USPAT	2003/10/01 13:12
-	490	(clipboard or (clip adj board) or "clip-board") and microsoft	USPAT	2003/10/01 13:14
-	408	((clipboard or (clip adj board) or "clip-board") and microsoft) and storage	USPAT	2003/10/01 13:14
-	151	(((clipboard or (clip adj board) or "clip-board") and microsoft) and storage) and temporary	USPAT	2003/10/01 13:14
-	44	((((clipboard or (clip adj board) or "clip-board") and microsoft) and storage) and temporary) and volatile	USPAT	2003/10/01 13:15

09/583520

US006589290B1

(12) **United States Patent**
Maxwell et al.

(10) **Patent No.: US 6,589,290 B1**
(45) **Date of Patent: Jul. 8, 2003**

(54) **METHOD AND APPARATUS FOR POPULATING A FORM WITH DATA**

(75) Inventors: **Duane Maxwell**, Coronado, CA (US); **William von Reis**, San Diego, CA (US); **Geoffrey D. Scott**, San Diego, CA (US)

(73) Assignee: **America Online, Inc.**, Dulles, VA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/430,535**

(22) Filed: **Oct. 29, 1999**

(51) **Int. Cl.**[7] **G06F 17/24**
(52) **U.S. Cl.** **715/507**
(58) **Field of Search** 715/505, 506, 715/507

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,794,259 A		8/1998	Kikinis	
5,799,115 A	*	8/1998	Asano et al.	382/305
5,974,430 A	*	10/1999	Mutschler et al.	707/505
6,088,700 A	*	7/2000	Larsen et al.	707/10
6,192,380 B1	*	2/2001	Light et al.	707/505
6,199,079 B1	*	3/2001	Gupta et al.	707/507
6,247,029 B1	*	6/2001	Kelley et al.	707/507
6,421,693 B1	*	7/2002	Nishiyama et al.	707/507
6,499,042 B1	*	12/2002	Markus	715/507

OTHER PUBLICATIONS

WO9946701A (Amazon.com; Gupta A.; Rajaraman A.) Sep. 16, 1999; p. 9, line9–line27; figs, 1D, 3C, 3D;p11, line 16–p12 line21; relevant to clms 1–33.

EP 0 918 424 A (IBM) May 26, 1999 (May 05, 1999); par. '0014; par. '0034; par. '0043; figure 12; relevant to Claims 1–33.

US 5,640,577 A (Scharmer Andrew J) Jun. 17, 1997 (Jun. 17, 1997); col. 5, line 12–col. 8, line 12; relevant to Claims 1–33.

* cited by examiner

Primary Examiner—Heather R. Herndon
Assistant Examiner—Charles A. Bieneman
(74) *Attorney, Agent, or Firm*—The Hecker Law Group

(57) **ABSTRACT**

A method and apparatus for populating a form with data is described. In one embodiment of the invention, a form is displayed to the user via the target application. Each form has one or more data receptacles. The data receptacles of a form are filled with data when the user executes a data population command. The form completion program executes the data population command when a graphical representation of a particular data set is placed over the form. Each data set is stored in an encrypted manner and is accessible to users who enter the appropriate information into an authentication mechanism. To populate a form with data the form completion program obtains an image of the form and then searches for a template file that resembles the form image to within a certain threshold. The template files are typically stored on the computer hosting the target application in a template directory that is arranged according to a predefined structure. The form completion program is configured to search for templates that resemble the form image in the template directory to within a certain threshold.

33 Claims, 10 Drawing Sheets

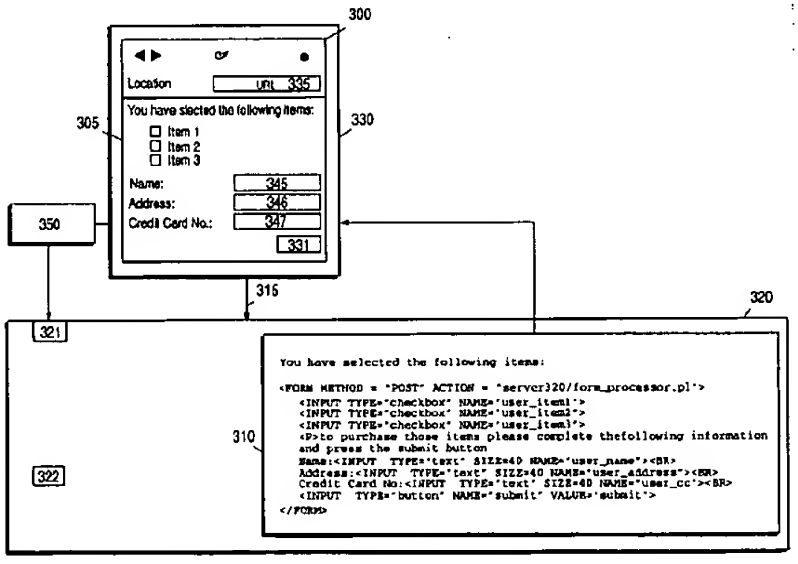


FIGURE 1

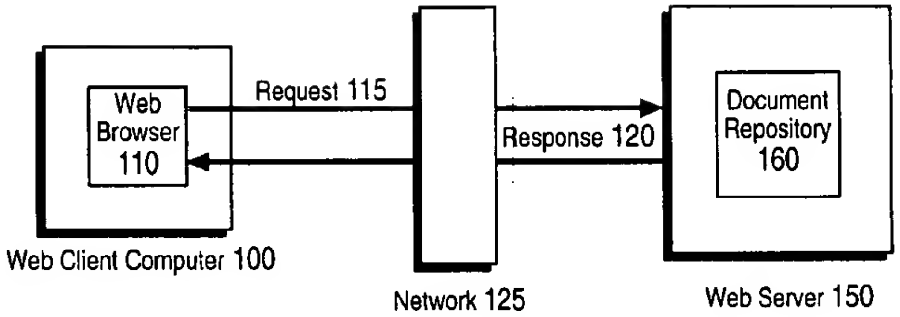
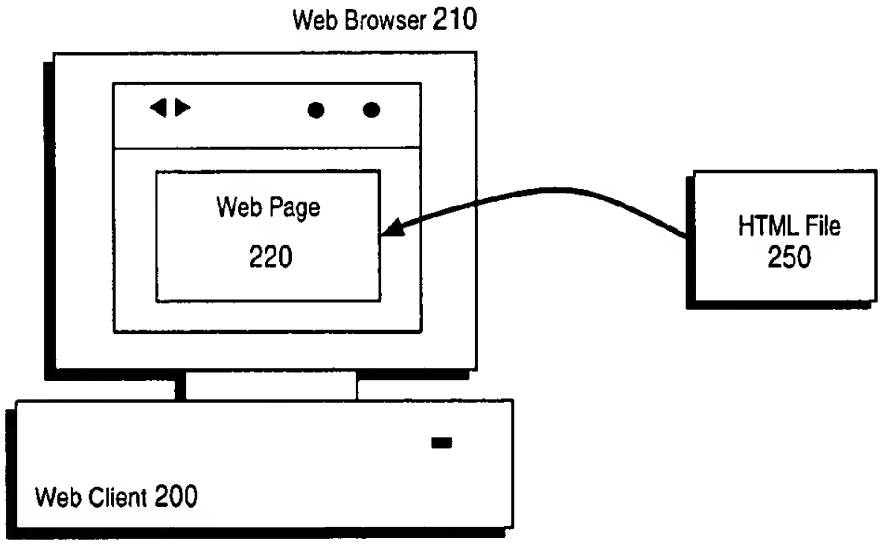


FIGURE 2



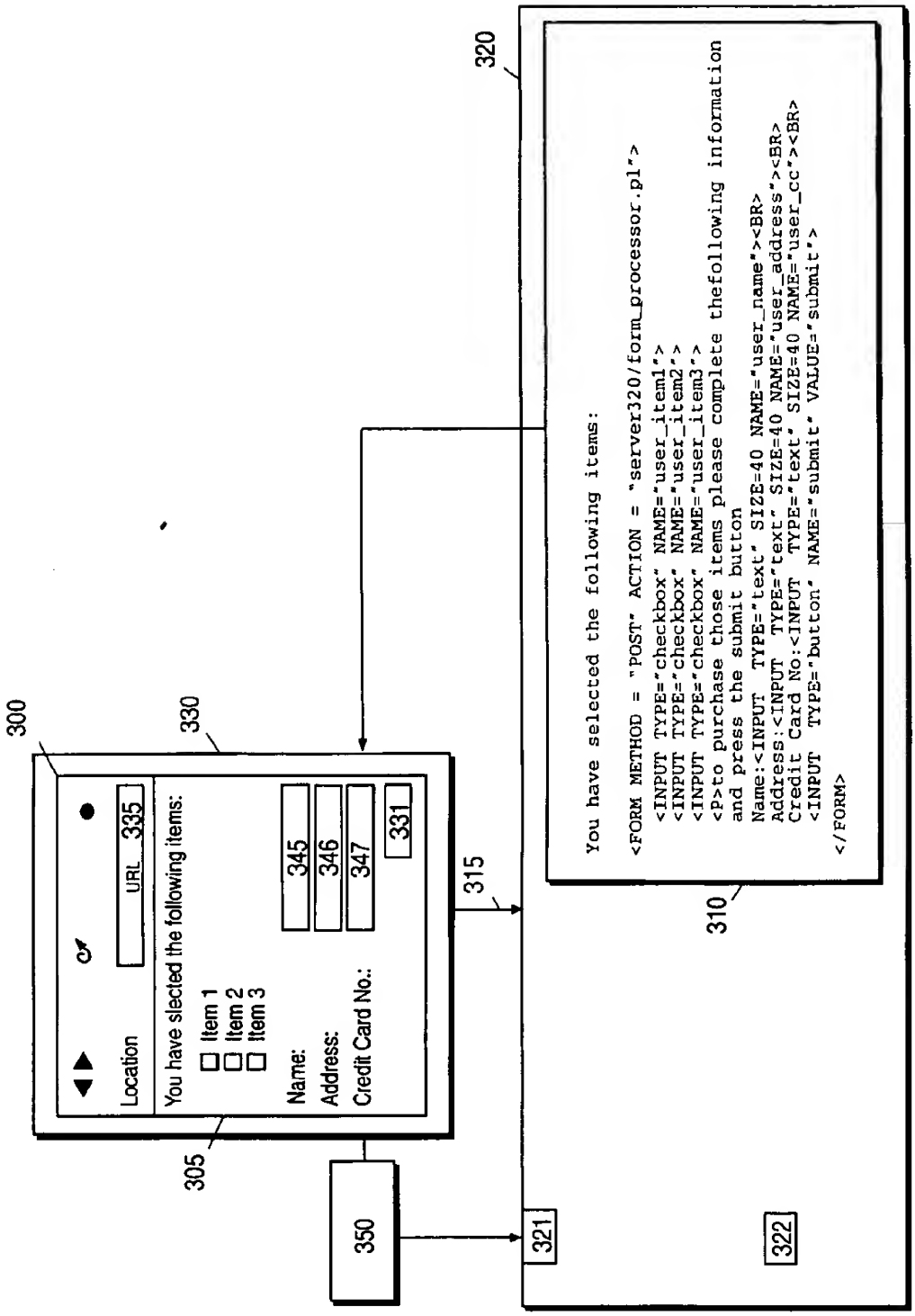

300
330
320
315
310
305
350
321
322
Location
URL 335
You have slected the following items:
Item 1
Item 2
Item 3
Name:
Address:
Credit Card No.:
345
346
347
331
You have selected the following items:
<FORM METHOD = "POST" ACTION = "server320/form_processor.pl">
<INPUT TYPE="checkbox" NAME="user_item1">
<INPUT TYPE="checkbox" NAME="user_item2">
<INPUT TYPE="checkbox" NAME="user_item3">
<P>to purchase those items please complete thefollowing information
and press the submit button
Name:<INPUT TYPE="text" SIZE=40 NAME="user_name">
Address:<INPUT TYPE="text" SIZE=40 NAME="user_address">
Credit Card No:<INPUT TYPE="text" SIZE=40 NAME="user_cc">
<INPUT TYPE="button" NAME="submit" VALUE="submit">
</FORM>

FIGURE 3

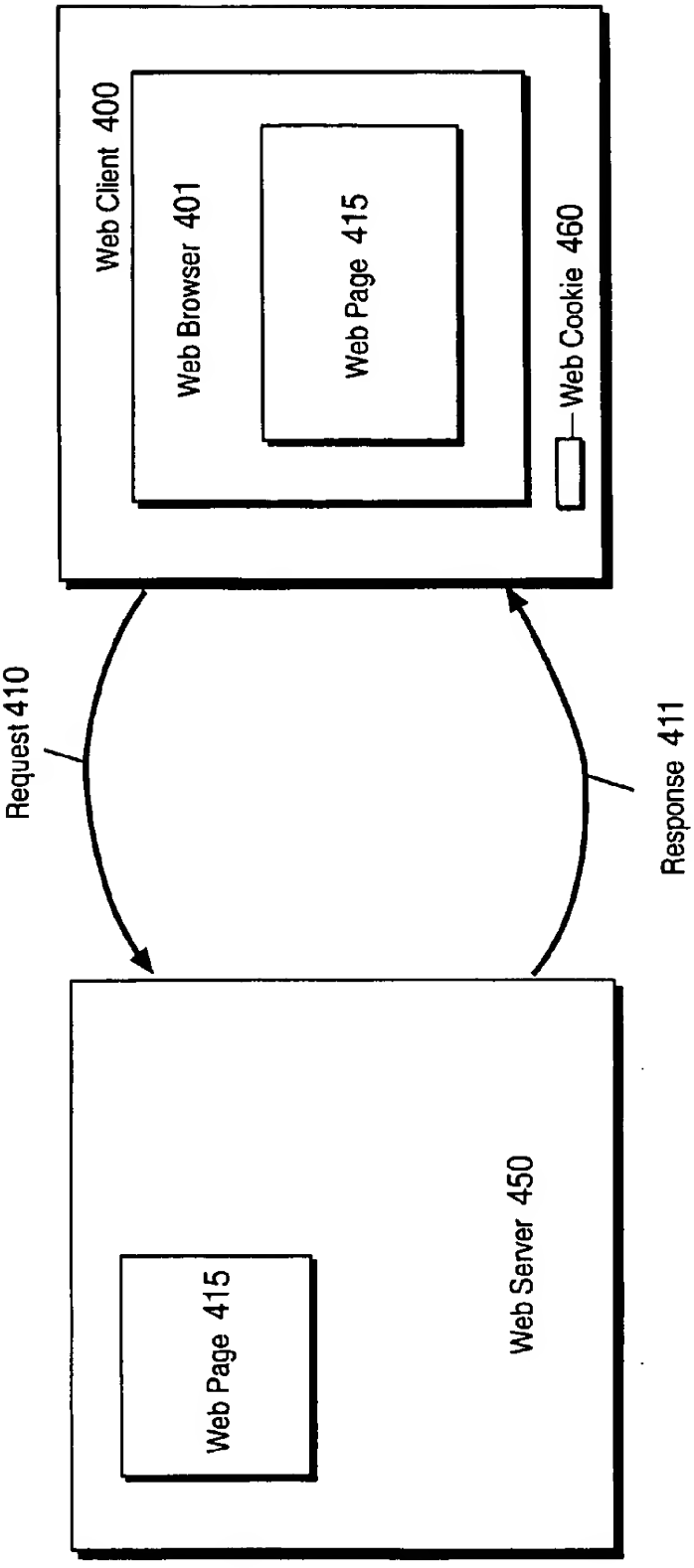

Web Client 400
Web Browser 401
Web Page 415
Web Cookie 460
Request 410
Response 411
Web Server 450
Web Page 415

FIGURE 4

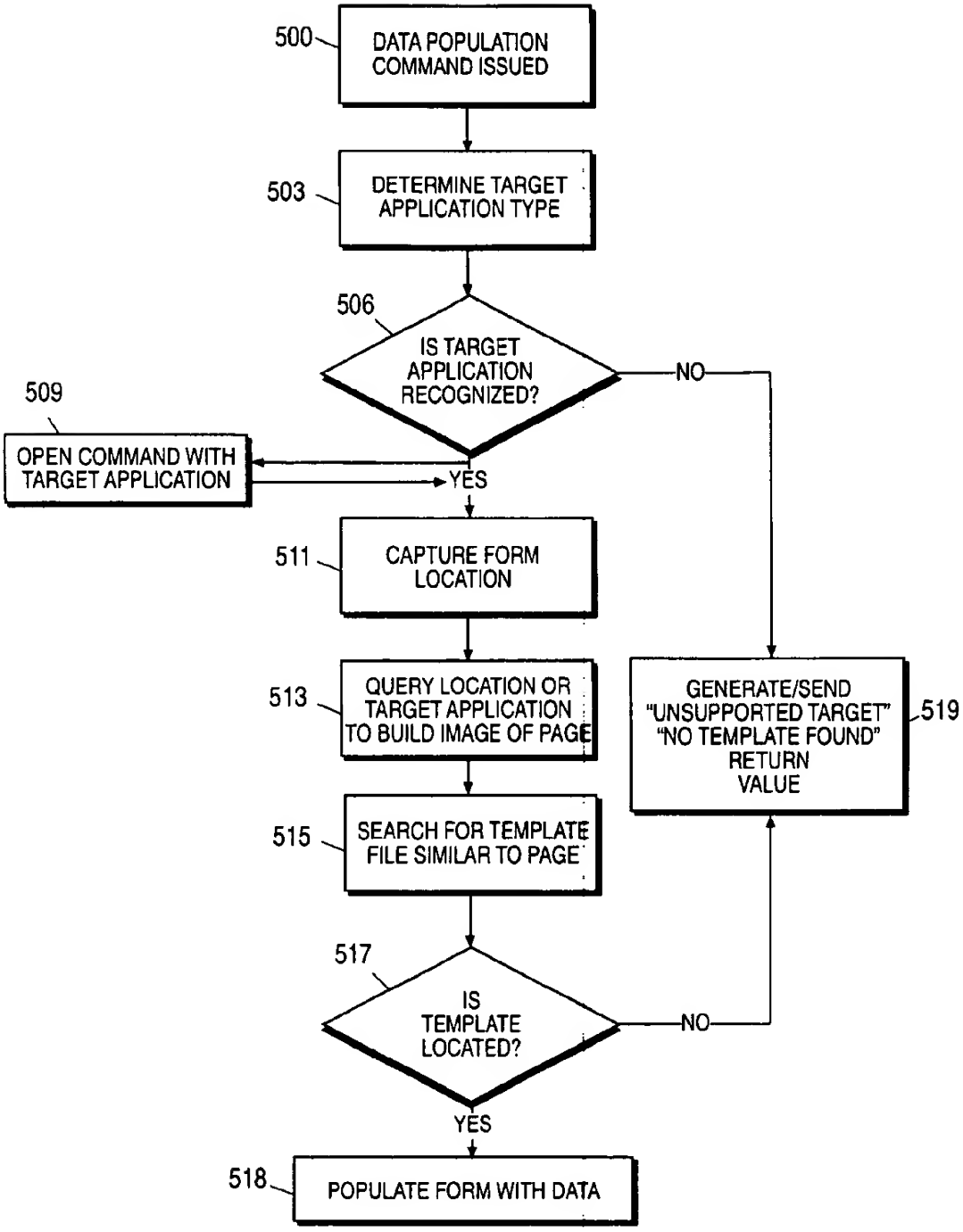

500
DATA POPULATION COMMAND ISSUED
503
DETERMINE TARGET APPLICATION TYPE
506
IS TARGET APPLICATION RECOGNIZED?
NO
509
OPEN COMMAND WITH TARGET APPLICATION
YES
511
CAPTURE FORM LOCATION
513
QUERY LOCATION OR TARGET APPLICATION TO BUILD IMAGE OF PAGE
515
SEARCH FOR TEMPLATE FILE SIMILAR TO PAGE
517
IS TEMPLATE LOCATED?
NO
YES
518
POPULATE FORM WITH DATA
519
GENERATE/SEND "UNSUPPORTED TARGET" "NO TEMPLATE FOUND" RETURN VALUE

FIGURE 5

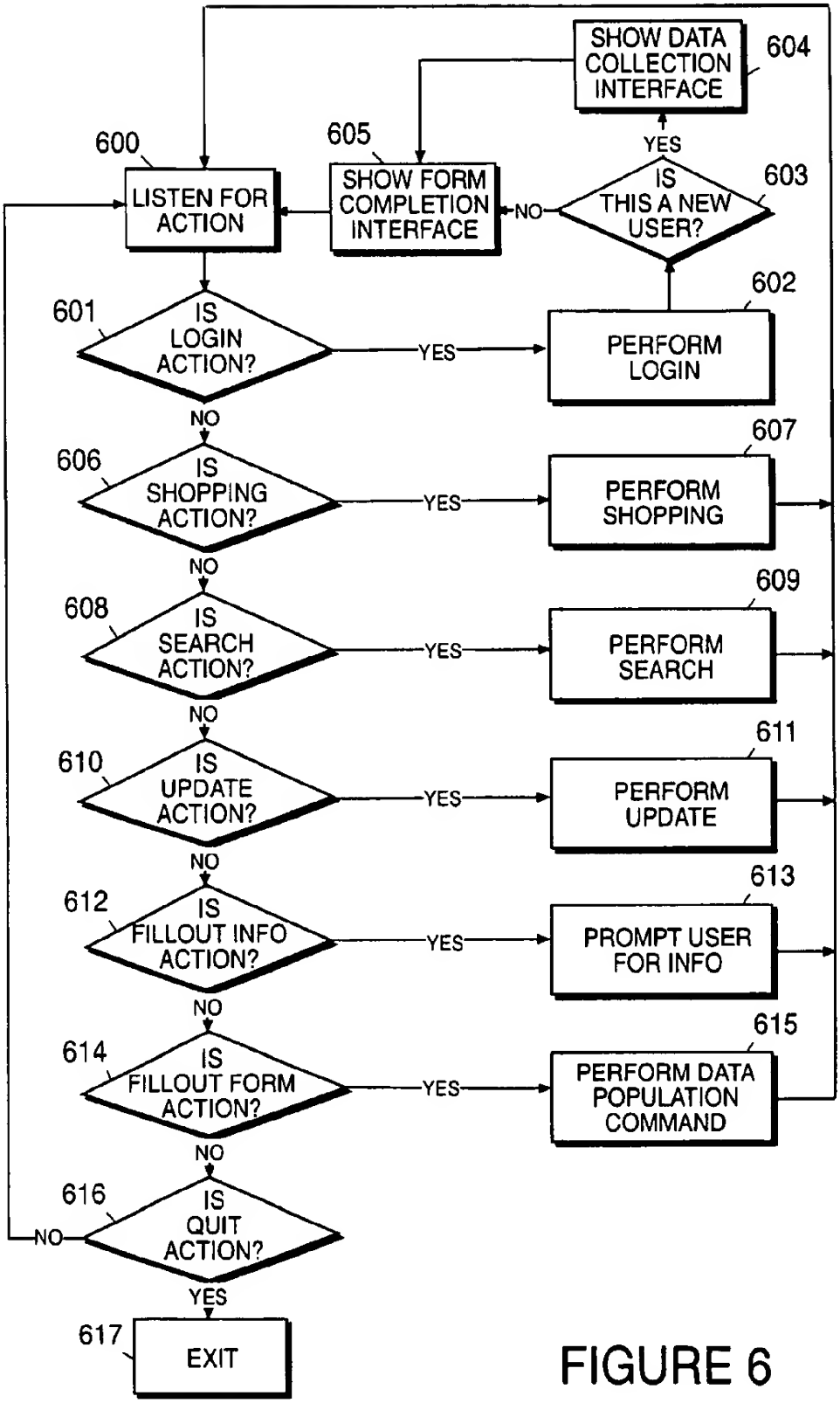

SHOW DATA COLLECTION INTERFACE
604
600
605
YES
LISTEN FOR ACTION
SHOW FORM COMPLETION INTERFACE
NO
IS THIS A NEW USER?
603
602
601
IS LOGIN ACTION?
YES
PERFORM LOGIN
NO
607
606
IS SHOPPING ACTION?
YES
PERFORM SHOPPING
NO
609
608
IS SEARCH ACTION?
YES
PERFORM SEARCH
NO
611
610
IS UPDATE ACTION?
YES
PERFORM UPDATE
NO
613
612
IS FILLOUT INFO ACTION?
YES
PROMPT USER FOR INFO
NO
615
614
IS FILLOUT FORM ACTION?
YES
PERFORM DATA POPULATION COMMAND
NO
616
IS QUIT ACTION?
NO
YES
617
EXIT

FIGURE 6

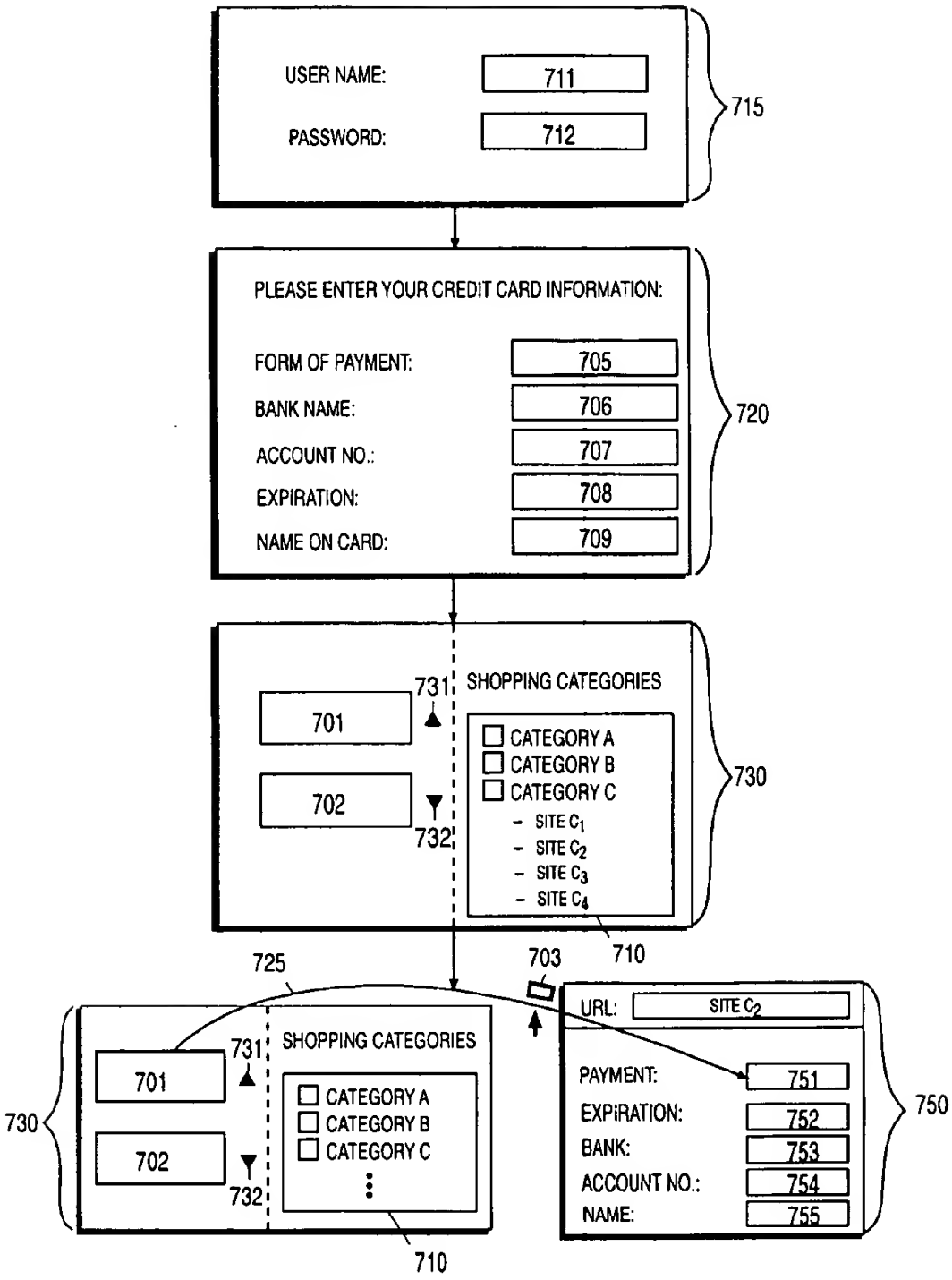

USER NAME:
711
PASSWORD:
712
715
PLEASE ENTER YOUR CREDIT CARD INFORMATION:
FORM OF PAYMENT:
705
BANK NAME:
706
ACCOUNT NO.:
707
EXPIRATION:
708
NAME ON CARD:
709
720
731
SHOPPING CATEGORIES
701
702
732
CATEGORY A
CATEGORY B
CATEGORY C
- SITE C1
- SITE C2
- SITE C3
- SITE C4
730
710
725
703
URL:
SITE C2
PAYMENT:
751
EXPIRATION:
752
BANK:
753
ACCOUNT NO.:
754
NAME:
755
750

FIGURE 7

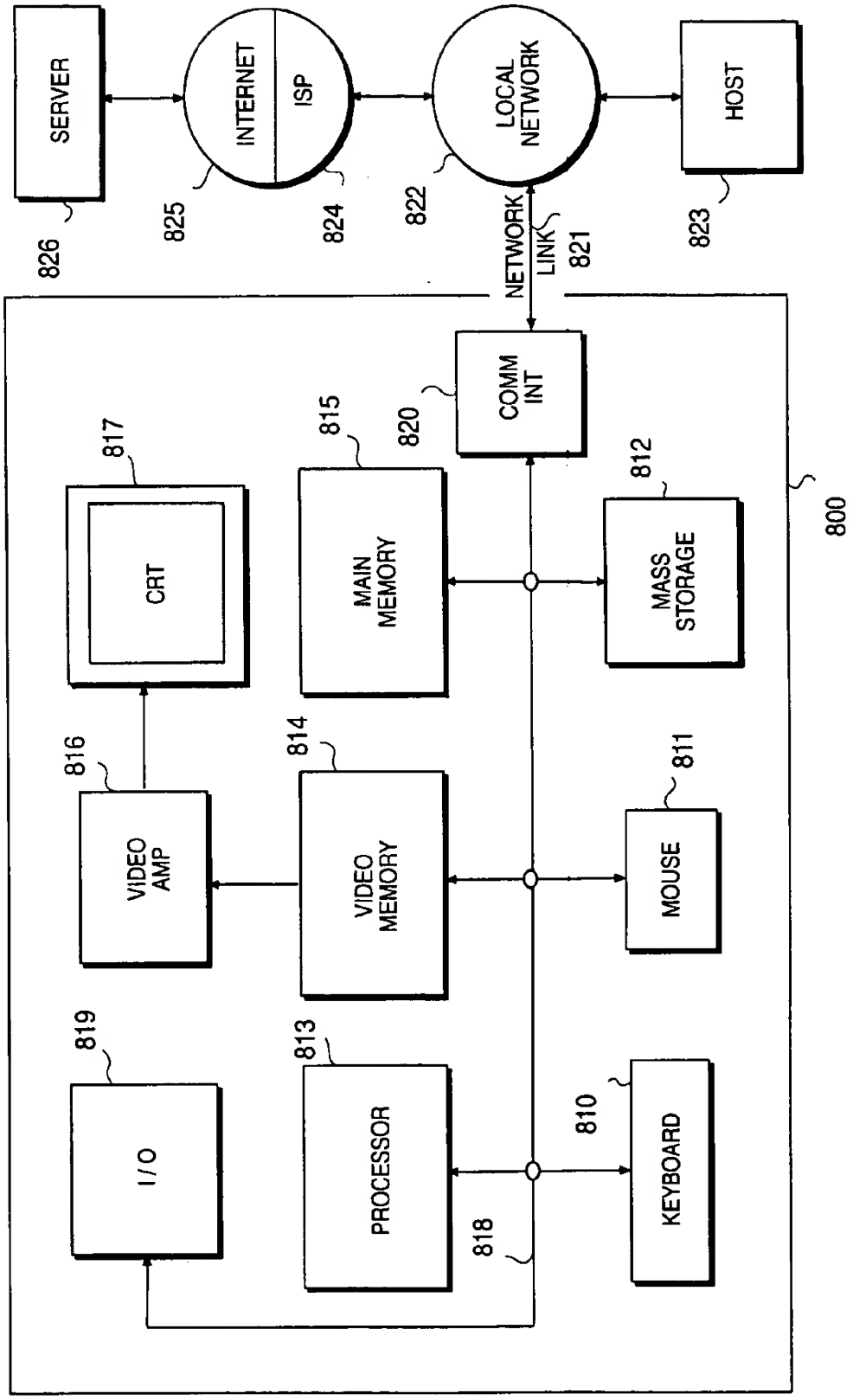

SERVER
826
INTERNET
ISP
825
824
LOCAL NETWORK
822
HOST
823
NETWORK LINK
821
800
COMM INT
820
MAIN MEMORY
815
MASS STORAGE
812
CRT
817
VIDEO AMP
816
VIDEO MEMORY
814
MOUSE
811
I/O
819
PROCESSOR
813
KEYBOARD
810
818

FIGURE 8

Figure 9

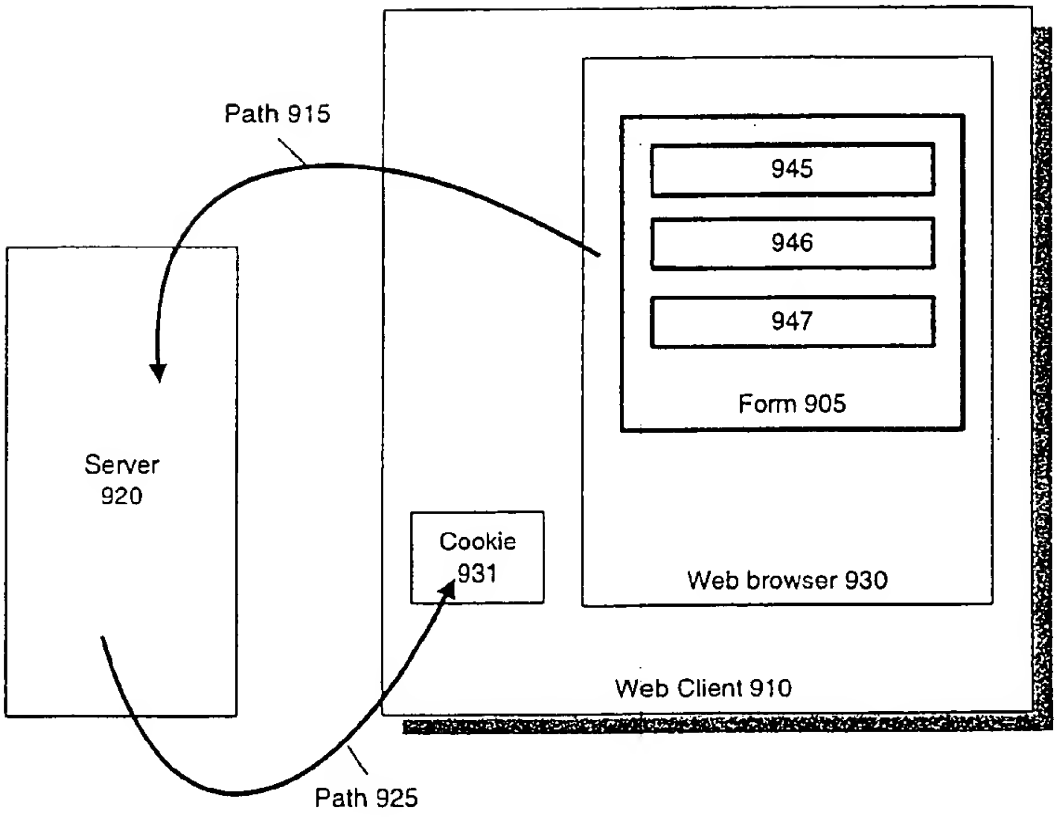

Path 915
945
946
947
Form 905
Server
920
Cookie
931
Web browser 930
Web Client 910
Path 925

Figure 10

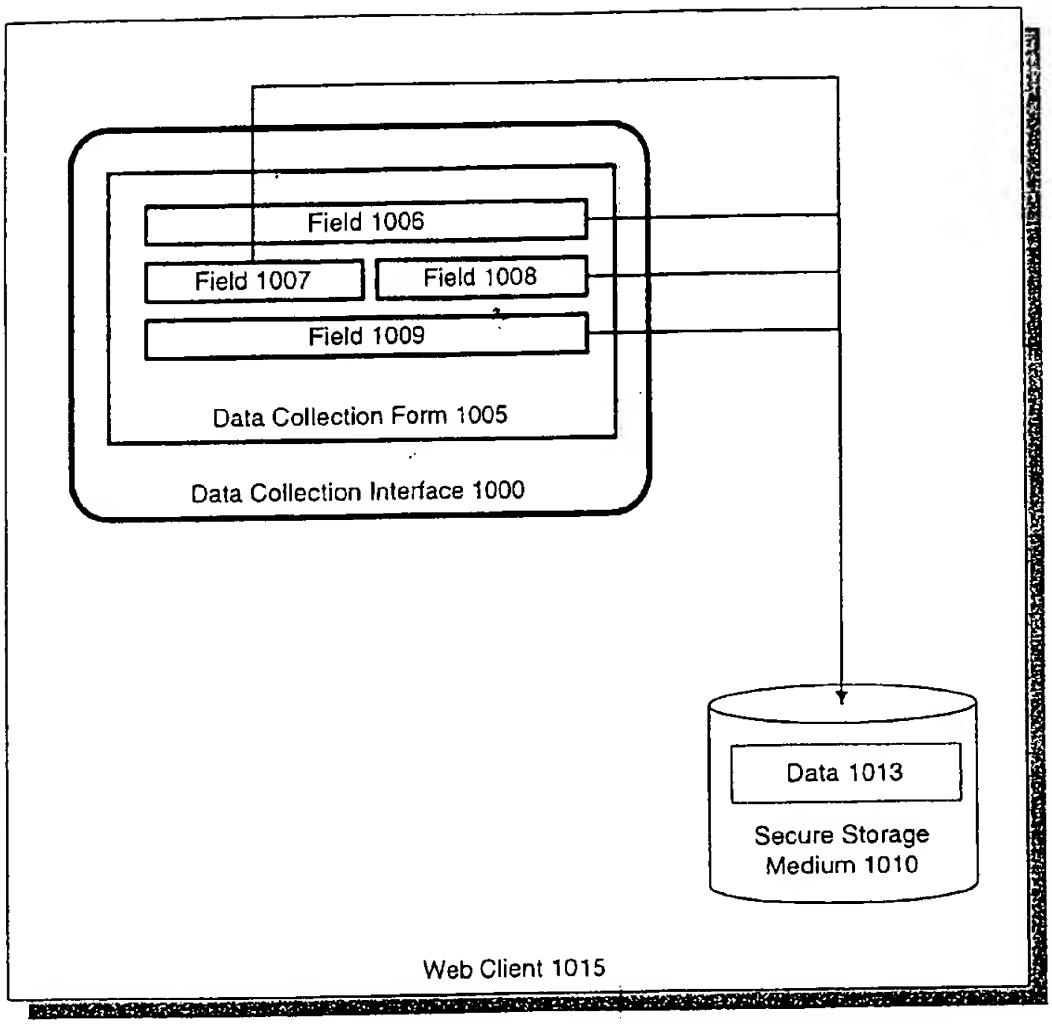
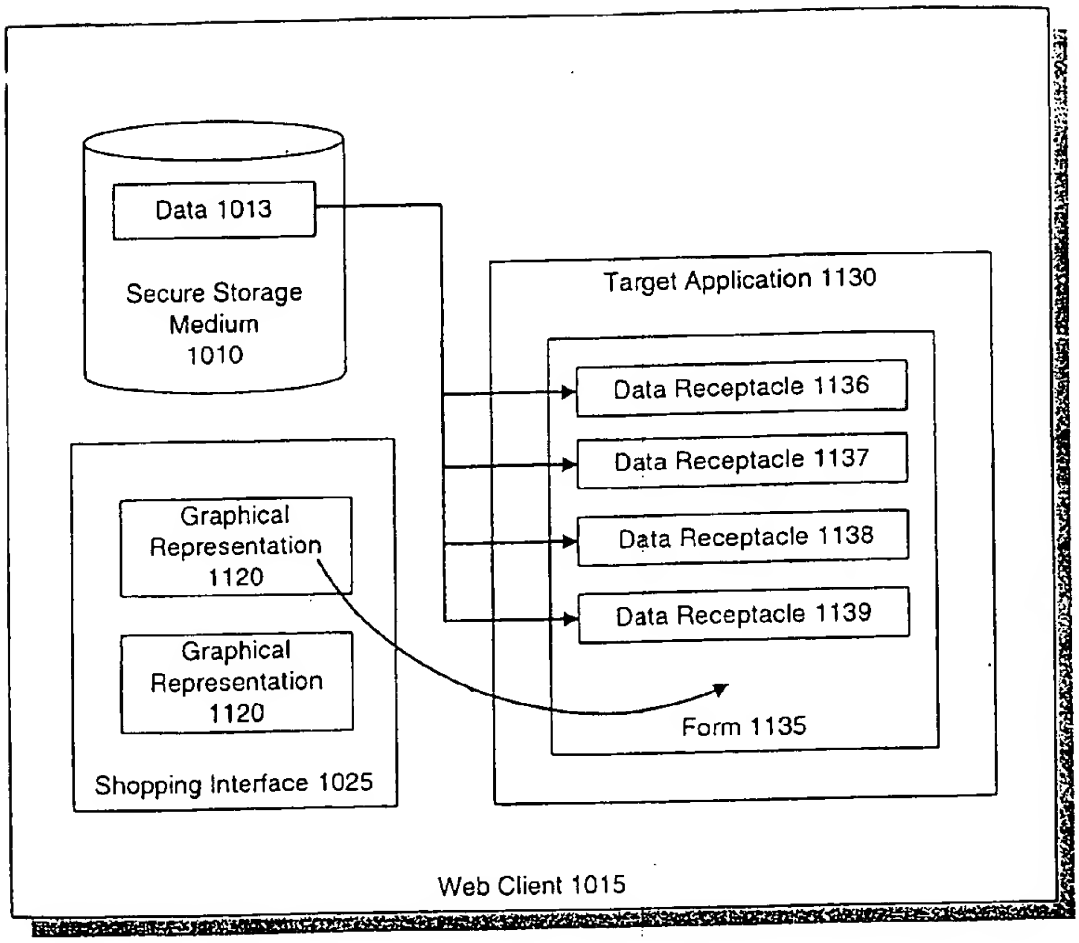

Field 1006
Field 1007
Field 1008
Field 1009
Data Collection Form 1005
Data Collection Interface 1000
Data 1013
Secure Storage Medium 1010
Web Client 1015

Figure 11


Data 1013
Secure Storage Medium 1010
Target Application 1130
Data Receptacle 1136
Data Receptacle 1137
Data Receptacle 1138
Data Receptacle 1139
Graphical Representation 1120
Graphical Representation 1120
Form 1135
Shopping Interface 1025
Web Client 1015

METHOD AND APPARATUS FOR POPULATING A FORM WITH DATA

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to computer networks and more particularly to a method and apparatus for populating a form with data.

2. Background Art

Computer users can shop for merchandise using the Internet. A number of companies maintain web sites where customers can purchase books, records, videos, and many other products. Other web sites provide subscription services that provide the user with access to a service or product. Typically, users make purchases or sign up for services by completing forms. A problem with current Internet schemes is that each field on a form must be filled out separately and manually. This is a time consuming and often frustrating process for the user.

Current systems do not provide a way for users to quickly complete a form (referred to as "populating" a form) with data. For example, when a user wishes to purchase a product online, the user typically supplies a credit card number, a shipping address, and any other personal information that may be needed to complete the transaction. The user supplies this information using an input device such as a keyboard to manually enter data into the fields of a form. However, after buying goods from one vendor, for example, the user may wish to purchase goods from a second vendor. The user must manually complete the second vendor's form even though the second vendor may require the purchaser to provide the same kind of information the user provided to the first vendor. This presents a problem to the user because it forces the user to manually enter the same information into different vendor's forms multiple times.

Another problem presented by the prior art is that users do not have a way to securely store sensitive information and then later use that information to populate a form. Current systems do not provide a way to prevent unauthorized users from obtaining access to sensitive information stored on a user's computer. Instead such systems leave data that is entered into a form readily accessible. For example, data that is entered into a form may be stored locally using a technology called cookies. A cookie is a local representation of information related to a particular web page that was previously visited by the user. Cookies are not encrypted and may be read by anyone who can obtain them. The term anyone also includes any computer program that knows where the cookies are stored. This is problematic because it unnecessarily exposes sensitive information to unauthorized users. Therefore, there is a need for a method and apparatus for populating forms with data where that data is accessible only to authenticated users.

The problems associated with form population may be better understood by the following discussion of the Internet/World Wide Web, web page creation, embedded forms, cookies, and web browser technology.

The Internet/World Wide Web

A web browser is a type of computer program that provides users with a mechanism for accessing the World Wide Web (WWW). The WWW is a segment of the Internet comprised of numerous web clients and web servers that communicate with one another using a standard set of protocols. A web server is a computer configured to provide web pages to the web client upon request. A web client typically utilizes the web browser application to request web pages from the web server.

The Internet is a global computer network comprised of an amalgamation of interconnected networks that are capable of readily communicating with one another using a standardized set of protocols. Protocols provide a uniform set of communication parameters that enable computers to effectively transmit and receive data. Most computer networks, including the Internet, utilize several different layers of protocols. Each layer provides the network with different functionality.

The WWW is a segment of the Internet that utilizes an application layer protocol called the HyperText Transfer Protocol (HTTP) to disseminate and to obtain information from users. HTTP is a request/response protocol used with distributed, collaborative, hypermedia information systems. In operation, HTTP enables one computer to communicate with another. For example referring now to FIG. 1, web client **100** can use HTTP to communicate with web server **150** via network **125**. In this scenario the web server acts as a repository for files **160** and is capable of processing the web client's requests for such files. The files **160** stored on the web server may contain any type of data. For example, the files may contain data used to construct a form, image data, text data, or any other type of data.

HTTP has communication methods that allow web client **100** to request or send data to web server **150**. The web client **100** may use web browser **110** to initiate request **115** and receive one or more files from file repository **160**. Typically, web browser **110** sends a request **115** for at least one file to web server **150** and the web server forwards the requested file to web client **100** in response **120**. The connection is then terminated between web client **100** and web server **150**. Web client **100** then uses web browser **110** to display the requested file. A client request **115** therefore, consists of establishing a connection between the web client and the web server using network **125**, issuing a response **120** to the request **115**, and terminating the connection.

Web server **150** does not utilize state information about the request once the connection is terminated. HTTP is, therefore, a stateless application protocol. That is, a web client can send several requests to a web server, but each individual request is treated independent of any other request. In some instances, the web server maintains a record of such requests, but the web server does not use that information to process later requests. Thus, for example, if a form is completed by the user and submitted to the web server for processing, the web server may maintain a record of the data entered into the form, but that record will not be used to later influence another request from the same client. In other words, web server **150** may record a request in a log file, but it does not later read from the log to determine how to respond to another request from the same client.

Once a file is sent from web server **150** to web client **100** it becomes ready for display. The web client's **100** web browser **110** is typically used to format and display files. Web browser **110** allows the user to request and view a file without having to learn a complicated command syntax. Examples of several widely used web browsers include Netscape Navigator, Internet Explorer, and Opera. Some web browsers can display several different types of files. For example, files written using the HyperText Markup Language (HTML), the JavaScript programming language, the ActiveX programming language, or the Portable Document Format (PDF) may be displayed using a web browser. It is also possible to display various other types of files using language such as Standard Generalized Markup Language (SGML) or eXtensible Markup Language (XML).

Creating a Web Page

A form, which provides one or more places for a user to enter data, can be embedded inside of a web page. A web page may be created using a variety of different data formats and/or programming languages. Most web pages, and as a result most forms, are created using the HyperText Markup Language (HTML). The techniques used to create a web page will now be discussed in further detail.

HTML is a language that may be used to specify the contents of a web page (e.g. web page **220**). An HTML description is typically comprised of a set of markup symbols which are described in more detail below. HTML file **250** or any type of data file that contains the markup symbols for web page **220** may be sent to web browser **210**. Web browser **210** executing at web client **200** parses the markup symbols in HTML file **250** and produces web page **220**, which is then displayed, based on the information in HTML file **250**. Web page **220** may contain text, pictures, or forms comprised of embedded text fields, checkboxes, or other types of data that is to be displayed on the web client using web browser **210**. Consequently, HTML document **250** defines the web page **220** that is rendered by web browser **210**. For example, the following set of markup symbols directs web browser **210** to display a title, a heading, and an image called "image.jpg":

```
<HTML>
<HEAD>
<TITLE> This is a document title </TITLE>
</HEAD>
<BODY>
<H1> This text uses heading level one </H1>
<IMG SRC="http://www.idealab.com/image.jpg">
</BODY>
</HTML>
```

In the above example, markup symbols (e.g. "<" and ">") indicate where each HTML command (e.g. TITLE) begins and ends. An HTML command, which is typically surround by markup symbols, provides the web browser with instructions to execute. Markup symbols typically surround an HTML command. The "<" symbol indicates the start of an HTML command an the "<" symbol indicates the end of an HTML command. Each start or end command has a corresponding ">" to indicate the close of that particular command. Information associated with the HTML command may be contained within the HTML command's start and end symbols. An HTML command is used by the web browser **210** to determine how to process the block of information associated with the two commands.

In the above example, "<TITLE>", and "</TITLE>" are examples of HTML commands surrounded by markup symbols. The "</TITLE>" HTML command directs web browser **210** to place the text "This is a document title" in the title bar of web browser **210**.

Some HTML commands have attribute names associated with the command. For example, HTML command "<IMG>", directs web browser **210** to display an image. A "SRC=" attribute identifies the location and name of the image to be displayed. In the above example, the statement "<IMG SRC="http://www.idealab.com/image.jpg">" tells the web browser to display an image named "image.jpg" that can be obtained from the web server located at "http://www.idealab.com."

Embedding a Form into a Web Page

An HTML file may also contain HTML commands that cause the web browser to render a web page that contains fields for entering data. The portion of the web page that contains the data entry fields may be referred to as a form (e.g. the portion enclosed in the HTML tags <FORM></FORM>). When the web page is comprised of primarily data entry fields the entire web page is sometimes also referred to as a form. As is discussed below, HTML includes an HTML form command that may cause the browser to display data entry fields. A text box, a drop down menu, a check box, a command button, a toggle button, or any other kind of interface component capable of receiving input are some examples of the type of data entry fields that may be placed in a form. Data is manually entered into the fields by the user and then submitted to a web server for processing. As previously discussed, when a user wishes to use a form to purchase a product, for example, the user manually fills in text boxes located on the form with the type of information needed to carry out the transaction. Other types of data entry fields require the user to select from one or more options such as in the case of a menu, a toggle button, or a radio button, for example. When the form is complete the user submits the completed form to the vendor's web server by pressing a command button, for example.

A problem with collecting information from the user in this manner is that the user is required to manually enter data into the fields of a form. This takes time and becomes unnecessarily burdensome when the user wishes to complete more than one form with the same information.

FIG. **3** provides an example of, a form created using the HTML definition language. Code block **310** contains HTML command examples. When a document comprising code block **310** is transmitted to web browser **300** executing on web client **330**, it causes form **305** to be displayed. Web browser **300** displays form **305** by parsing the HTML commands contained in code block **310** and then using the information obtained to format form **305**. Once the user finishes entering information into form **305**, the user may submit the form by pressing command button **331**. Pressing command button **331** sends the information entered by the user in form **305** to web server **320** using the POST method. However, one of ordinary skill in the art could modify code block **310** to use the GET method instead of the POST method.

The <FORM>command shown in code block **310** indicates the beginning of a form. Once the initial FORM command is placed into the HTML document other HTML commands may be entered between the initial FORM command and the closing FORM command (e.g. </FORM>) that represent, for example, one or more data entry fields such as a text-box, drop-down lists, check boxes, radio buttons, and input buttons. The INPUT command is used to specify different types of data entry fields. The type of data entry field created is dependant upon the value assigned to the INPUT command's TYPE attribute. For example, the INPUT command can create a text-box, a password field, a hidden field, a button, a radio button, or a checkbox. A "text" value assigned to the TYPE attribute of an INPUT command creates a text box, for example. Similarly, a TYPE of "checkbox" creates a checkbox. The following code, if inserted after the FORM element, creates a text box and a checkbox:

```
<INPUT TYPE="text" NAME="user_name">
<INPUT TYPE="checkbox" NAME="user_item1">
```

The HTML tags and text contained in code block **310**, for example, create form **305** when displayed using web browser **300**.

A <FORM>tag may contain an ACTION and/or METHOD attribute. The ACTION attribute specifies what program on the server to execute when form **305** is submit-

ted for processing. The METHOD attribute describes how data entered into the form is handled. There are two METHOD choices: GET and POST. Each one of them is capable of processing the data entered into a form. To illustrate, if a user named Bill who resides at 130 West Union Street, Pasadena, Calif. 91103 with a credit card number of 1111222233334444 completes the Name Address, and Credit Card number fields shown on form **305** by filling in spaces **345–347**, then the following data string **350** is created and sent to web server **320**.

user_name=Bill&user_address=130 West Union Street Pasadena CA 91103&user_cc=1111222233334444

This data string may be handled by either the GET method or the POST method. However, in FIG. 3 it is handled by the POST method. The GET method appends information entered into form **305** by the user onto the end of a Uniform Resource Locator (URL **335**) or in a QUERY_STRING environment variable. URL **335** is submitted to web server **320** by web client **330** in an HTTP request **315**. Web server **320** is responsible for processing the submitted information. A problem with the GET method is that the information contained in URL **335** is not stored on web client **330** for later use and cannot therefore be later used to populate form **305**. With the POST method, the information placed into form **305** by the user is placed into the data stream of the HTTP protocol. This allows web server **320** to process form **305** data using the standard input data stream to a separate program or script on the server. Standard input is a method for providing a program or script with data that is well know to programmers skilled in the art. The POST method also does not provide a way to store data on web client **330** for later use.

The ACTION attribute of the FORM tag indicates where the computer program tasked with processing data string **350** resides. For example, the following portion of code block **310** specifies that a computer program (e.g. a PERL script) named form_processor.pl is located in cgi-bin **321** of sever **320**.

<FORM METHOD=POST ACTION=http://server320/form_processor.pl>

The form_processor.pl program is responsible for processing data string **350**. Programs that utilize the Common Gateway Interface (CGI) standard are typically located in a common directory (e.g. cgi-bin **321**). CGI is a specified standard for communicating between HTTP servers and server-side gateway programs. A CGI program is capable of storing data supplied by the user, but it cannot determine whether the user asking for the form is the same user that previously submitted data to the CGI program. If the user has never visited the web site before, the CGI program does not have any information about that user. CGI programs can collect information about users who visit a web site, but such programs cannot collect information about users who have never visited the web site.

The program "form_processor.pl" used in the above example is an example of a server-side program that processes data input by the user via form **305**, for example. When form **305** is submitted web server **320** executes the "form_processor.pl" program and passes the data that was entered by the user and then sent from web client **330** to server **321**. That is, data string **350** is passed to "form_processor.pl" when a form is submitted. When the gateway program is done processing data string **350** the result is sent to server **320** and a response may be forwarded back to web client **330**. Gateway programs can be compiled programs written in languages such as C or C++ or they can be executable scripts written using a scripting language such as PERL, TCL, or various other language. Once data string **350** is processed it may be stored on server **320** in data store **322**. A problem with using CGI programs is that since they are located on the server, information cannot easily be saved on the web client for later use.

Thus, existing mechanisms for processing forms reside on the server and are unable to populate a form on the client with user data from users that have never visited the web site having the form (e.g. the data is only available to web servers the user has previously visited).

HTTP Cookies

One example of data that may be stored on the client is a cookie. A cookie is an HTTP header that consists of a text-only string. The text-string is entered into the memory of a web client and accessible by the web browser. This text-string is initially set by the computer serving the web page and consists of the domain name, path, lifetime, and value of a variable that the server sets. If the lifetime of this variable is longer than the time the user spends at that site, then the text-string is saved on the web client for later use. As a result, cookies provide a way to store and later recall values entered into a web page by the user. For example, cookies allow a web server to individually customize a web page for each user who visits the page.

A cookie may be used to populate the elements of a form that a user has previously completed. For example, referring now to FIG. **9**, if a user completes field **945–947** of form **905** and submits it to server **920** via submission path **915**, server **920** can place the data submitted by the user in cookie **931** by sending web client **910** data via path **925**. If the user returns to form **905** again Web browser **930** executing at web client **910** may use the data saved in cookie **931** populate fields **945–947** with the data previously entered by the user. Thus, a cookie can free the user from having to complete form **905** more than once. Cookies can be used to fill forms the user has previously visited. However, the data that is stored by the cookie cannot be used to populate other forms. That is, a problem with cookies is that forms located on different servers are not provided access to the same cookie. Only web servers that are listed as having permission to access a particular cookie may obtain such access.

A cookie is only accessible to the server that set, or created it. Thus, a second form cannot use the first form's cookies to obtain the user's name and address. For example, the web server located at www.merchantA.com may be allowed to read a cookie, but the web server located at www.merchantB.com may not be allowed to read the cookie created by www.merchantA.com

Referring now to FIG. **4**, the process used to create and retrieve a cookie is illustrated. Initially, web browser **401** which resides on web client **400** issues a request **410** for web page **415** from web server **450**. Web server **450** responds by sending a copy of web page **415** to web client **400** via response **411**. Web client **400** uses web browser **401** to display web page **415** to the user. To create a cookie **460**, web server **450** sends a "Set-Cookie" command in the header line contained in response **411**. For example, in response to a request **410**, web server **450** may send the following command in HTTP response **411**:

Set-Cookie: NAME=VALUE; expires=DATE; path=PATH; domain=DOMAIN_NAME; secure.

The NAME and VALUE parameter contain the information included in the cookie. DATE is the time at which the cookie expires and is thus no longer saved on web client **210**. DOMAIN is the host address or domain name for which the cookie is valid. For example, if a cookie is set by a computer having a domain name of www.merchantA.com, only the

server responsible for that particular domain name is provided access to the cookie. The PATH parameter specifies a subset of URL's at the appropriate domain for which the cookie is valid. If the keyword secure is used then the cookie is only transmitted over a secure connection. All of these parameters except the NAME=VALUE are optional to set a cookie. Once the Set-Cookie command is sent to web client **400** a cookie **460** is placed on web client **400**.

To create a cookie **460** using the UNIX operating system, for example, web server **450** could execute the following shell script in response to a request **410**:

```
#!/bin/sh
echo Content-type: text/html
echo Set-cookie: FooBar=foo; expires=Wednesday,
  02-11-99 12:00:00 GMT
echo
echo <H1>A Cookie named Foobar containing the text
  foo is now present on your computer</H>
```

This script stores "FooBar=foo" on web client **400**. The following script allows web server **450** to read the HTTP cookie:

```
#!/bin/sh
echo Content-type: text/html
echo
echo The data supplied here was obtained from a
  cookie:<P>
echo $HTTP_COOKIE
```

Once cookie **460** is created, the information stored in the cookie can be used in many different ways. Information in cookie **460** may be accessed by a script that has authorization to insert cookie **460** data into a form, for example. However, a problem with cookies that no authentication mechanism is required to obtain the information they store. A cookie is a text file and is not stored in encrypted form. While a browser may limit access to a cookie, the text that is stored in the cookie may be accessed or read by any program that can read a text file. Therefore, it is not wise to store any sensitive data in a cookie. An additional problem with cookies is that users do not have to enter a user name and password to obtain access to the cookie.

Predefined Paste Operation

Some web browsers provide users with a mechanism for pasting predefined data into a form. Under the generic preferences menu of the Opera web browser, for example, the user can select a command button labeled personal information. When this command button is selected a dialog box that contains fields for entering personal information opens. If the user desires, the user can enter a name, an address, a phone number, an e-mail address, and/or any other kind of information into the text fields of the dialog box.

Once the user enters information into the fields of the dialog box and clicks "OK" the information that was entered can be pasted into the fields of a form by performing a right click operation. For example, if the user encounters a form that has a place for entering an address, the user can elect to paste the address previously specified in the dialog box by first selecting the address field and then right-clicking on a pointing device such as a mouse and selecting from a drop down menu the item titled "insert address".

A problem with pasting information into the fields of a form in this manner is that the user is required to manually select the place to insert the information and the type of information to insert into that place. Another problem is that unauthorized users are not prevented from accessing the information entered into the dialog box.

SUMMARY OF THE INVENTION

A method and apparatus for populating a form with data is described. One embodiment of the present invention executes on a web client that is connected to a computer network such as the Internet. The web client is capable of obtaining web pages that contain forms from the Internet. The web client contains computer code, such as a form completion program, configured to enable a user to populate a form with data. The form completion program interacts with a target application (e.g. a web browser) installed at the web client to provide the user with a mechanism for filling out multiple forms. The form completion program, for example, provides the user with a mechanism to complete a form by dragging a graphical representation of a data set over to the form and then dropping it.

In one embodiment of the invention, a form is displayed to the user via the target application. Each form has one or more data receptacles. The data receptacles of a form are filled with data when the user executes a data population command. The form completion program executes the data population command when a graphical representation of a particular data set is placed over the form. The form completion program collects data from the user via a data collection interface. The data collection interface provides data entry fields for the user to enter a particular type of information (e.g. a data set). Each data set that is entered using the data collection interface is stored in an encrypted manner and is accessible to users who enter the appropriate information into an authentication mechanism.

To populate a form with data the form completion program obtains an image of the form and then searches for a template file that resembles the form image to within a certain threshold. The template files are typically stored on the computer hosting the target application in a template directory that is arranged according to a predefined structure. The form completion program is configured to search for templates that resemble the form image in the template directory to within a certain threshold. However, the invention also contemplates searching for templates at alternate locations. Templates, for example, may reside on any other computer that is accessible via a telecommunication medium such as a computer network.

The form completion program examines each template file in order to determine if one or more of the template files resembles the form image to within a certain threshold. If a template file that resembles the form image to within a certain threshold is located, then the form completion acknowledges that a match occurred. When a match occurs the form completion program utilizes the template file to identify what kind of data to insert into each of the form's data receptacles. For example, the template file allows the form completion program to determine which of the data receptacles contain personal information and which data receptacles contain payment information. Once the form completion program successfully identifies what kind of data to insert into each data receptacle the program begins to input the appropriate kind of data into the appropriate data receptacle.

The present invention contemplates the use of multiple techniques to insert information into the data receptacles. The technique used is largely dependent upon the type of target application used to display the form.

DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of the various components used by the World Wide Web.

FIG. 2 is a block diagram illustrating the components used to create a web page.

FIG. 3 is a block diagram illustrating how web client interacts with a web server to display a form.

FIG. 4 illustrates the process used to create and retrieve a cookie.

FIG. 5 is an illustration of the process used by one embodiment of the invention to populate a form.

FIG. 6 illustrates some of the actions the form completion program listens for when it is in the listening mode.

FIG. 7 is an example of several of the interfaces presented to the user by the form completion program.

FIG. 8 illustrates the execution environment of one embodiment of the invention.

FIG. 9 illustrates the process used by cookies to save form data.

FIG. 10 is a block diagram illustrating how data is collected from the user in one embodiment of the invention.

FIG. 11 is a block diagram illustrating how the shopping interface used by one embodiment of the invention interacts with a target application.

DETAILED DESCRIPTION

A method and apparatus for populating a form with data is described. In the following description numerous specific details are set forth in order to provide a more thorough understanding of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In other instances, well-known features have not been described in detail so as not to obscure the invention.

One embodiment of the present invention executes on a web client that is connected to a computer network such as the Internet. The web client is capable of obtaining forms from the Internet and contains computer code configured to allow a user to populate a form with data. In one embodiment of the invention, a form completion program is created to execute the invention.

The form completion program interacts with a target application (e.g. a web browser) installed at the web client to provide the user with a mechanism for filling out multiple forms. The target application displays forms to the user. The form completion program may have multiple interfaces that each provide the web client **1015** with additional functionality. For example, referring now to FIG. **10**, a data collection interface **1000** is shown. The data collection interface **1000** may be used to collect and store multiple data sets from multiple users. A data set is a collection of related information (e.g. payment information or personal information) that represents any kind of information likely to be used to complete a form.

In one embodiment of the invention, the data collection interface collects a data set via a data collection form **1005**. Data entered into the data collection fields **1006–1009** of the data collection form **1005** may be stored in secure storage medium **1010**. The data stored in storage medium **1010** is utilized to populate a form when the user issues a form population command. The data held in secure storage medium **1010** may be encrypted and reside on web client **1015** and/or any other computational device available via a network communication medium. A user is not permitted to access secure storage medium **1010** until properly authenticated.

Each data set stored in secure storage medium **1020**, may be associated with a graphical representation **1120**. Referring now to FIG. **11**, for example, a data set comprising payment information may be represented by a credit card graphic. A data set comprising personal information may be represented by an identification card graphic. A data set may be associated with any type of graphic. In one embodiment of the invention, the graphical representations of each data set are displayed to the user via a shopping interface **1125**. The shopping interface may be used to execute a form population command. When executed, a form population command completes form **1135**. Form **1135** is displayed via a target application **1103** such as a web browser or other computer program.

The form completion program executes the data population command when a graphical representation **1120** is moved from the shopping interface **1120** to form **1135**. The data population command fills data receptacles **1136–1139** with data taken from secure storage medium **1010**. For example, if the user encounters a form **1135** that prompts the user for personal information (e.g. name, address, and phone number), then the user could provide that information to form **1135** by dragging a graphic (e.g. graphic **1120**) that represents the user's personal information from form completion program to form **1135**. When the user drops the graphical representation **1120** of the information onto the form **1135**, the data population command places data into the appropriate place of the form by obtain the data from secure storage medium **1010**, identifying what type of form **1135** is to be populated, and then using this information to fill the fields **1136–1139** of form **1135** with data. In one embodiment of the invention, the type of form **1135** that is to be populated is identified by a template file. However, the manner with which a form is populated is dependent upon the type of form that is being populated.

The form completion program, for example, provides the user with a mechanism to complete a form by dragging a graphical representation of a data set over to the form and then dropping it. Anytime the user encounters a form the user may elect to populate the form by performing this drag and drop operation.

In one embodiment of the invention, the data receptacles of a form are filled with data when the user executes a data population command.

Creating a Form

A form may be created using HTML or any other programming language capable of generating a document that provides a place for the user to enter data (e.g. a data receptacle). For example, a spreadsheet, a word processing document, a document created using PDF, a database table, or any other type of document having one or more data receptacles may be referred to as a form. A data receptacle is a field or place for holding and/or receiving data.

Data Population Command

In one embodiment of the invention, the data population command is executed by a form completion program. The form completion program may be configured to collect data from the user and execute a data population command when the user performs a drag and drop operation. Other types of computer programs, however, may also collect data from the user and perform a data population command. For example, an operating system, a dynamic link library, a Java applet, or any other type of executable computer program may be configured to execute a data population command.

The form completion program aids in the process of collecting data from the user by prompting the user for information likely to be required to complete a form. For example, the user might be prompted to enter personal information (e.g. name, address, phone number, etc . . .) and payment information (e.g. American Express number, Visa card number, MasterCard number). Once information is gathered, the form completion program creates a graphic that is representative of the type of information collected.

For example, if the user provides the form completion program with payment information such as a Visa Card or MasterCard number, then a graphic is created that represents the form of payment information the user supplied. For example, an icon depicting a Visa Card, MasterCard or other such card might be displayed. If the user provides personal information, such as a name and address, then a graphic is created that identifies the data associated with that particular personal information. For example, an image resembling a drivers license or identification card might be displayed. Each graphical representation that is created is associated with a different data set. The user may elect to create additional data sets that have a corresponding graphical representation. The user may customize the type of data set collected and the graphics associated with that data set. A graphic may represent any kind of data. The form completion program utilizes the information the user supplies (e.g. the data set) to populate a form. For example, when a graphic that is representative of a particular type of data is dragged from the form completion program and dropped on a form the data population command is executed, and the form is completed without requiring more input from the user.

When the user executes the data population command, data is placed inside the data receptacles of the form on which the user executed the data population command. Multiple data sets/graphics may be used to populate the same form. For example, a graphic representative of a credit card may be used to populate the form with credit card data and a graphic representative of a user's personal information may be used to populate the form with personal information.

Various types of data may be placed into each data receptacle. For example, executing a data population command may insert text data, image data, and/or any other type of data into one or more data receptacles. In one embodiment of the invention, the data utilized to populate the form is supplied by the user and stored on the user's computer before execution of the data population command. However, it is possible to obtain data from other sources. For example, the data population command may obtain data from a memory device that is available via a communication medium such as a computer network. The user executes a data population command by moving a graphical representation of the data from a first location to a second location.

Referring now to FIG. **5**, an illustration of the process used by one embodiment of the invention to populate a form is shown. The user initiates a data population command at step **500** by performing a drag and drop operation. Once the user issues a data population command the form completion program proceeds to step **503** where an attempt is made to determine what type of target application is displaying the form that is to be populated. For example, the form completion program may determine that the user is attempting to populate a form displayed using the Internet Explorer web browser. The type of application displaying the form to the user is referred to as the target application.

When the target application type (e.g. Internet Explorer) is determined, then the form completion program proceeds to step **506**. At step **506**, the form completion program determines whether it recognizes the target application. The form completion program may be configured to support a variety of different target applications.

If the target application is recognized the form completion program proceeds to step **509** where it begins to communicate with the target application in a manner understood by the target application. If the target application is not recognized, then a return value is sent, at step **519**, and the form completion program informs the user that this type of target application is not yet supported. At step **509**, the form completion program is configured to interact with different target applications using a variety of different communication techniques. A form displayed using Internet Explorer may be dealt with in a different way than a form displayed using Netscape Communicator (e.g. a different interface protocol may be used). The COM interface, for example, is used to communicate with Internet Explorer 4. However, the present invention also contemplates the use of other interfaces and API's such as ActiveX or Javascript. The type of protocol utilized by the form completion program is dictated by the target application. The present invention contemplates the use of a protocol capable of communicating with the target application. It is possible to utilize a variety of different protocols to open a communication channel between the form completion program and the target application.

Once step **509** is complete, the communication channel between the form completion program and the target application is open. This allows the form completion program to proceed to step **511** where it captures the location of the form. For example, if the form is displayed using a web browser, the form completion program will capture the Uniform Resource Locator (URL) that identifies where the form is located. The URL provides the form completion program with the address information needed to determine where the form being populated was obtained. For example, the URL may indicate that the user obtained the form from another computer accessible via a communication medium such as a network. At step **513**, the form completion program builds an image of the form the target application displays to the user. In one embodiment of the invention, the form completion program builds the form image by querying the computer (e.g. the server) that originally supplied the form to the target application. However, the form completion program may build a form image by querying any computer that has a version of the form that is displayed to the user. For example, the form image could be built by contacting a mirror server instead of the original server. The form completion program may also build the form image by querying the target application about its form properties or the form completion program may elect to utilize a local version of the form that is displayed to the user. For example, the form completion program may utilize a cached version of the form to build a form image. The present invention contemplates various techniques for building a form image. It is possible to utilize any version of the form the user is attempting to populate in order to build the form image.

Once an image of the form the user is attempting to populate is produced, the form completion program proceeds to step **515** where it searches for a template file that resembles the form image. The template file comprises a collection of form descriptions. In one aspect of the invention, the template file is a text file that contains one or more form descriptions. Each form description is typically associated with a particular form. However, one description may represent multiple forms if the forms are substantially similar to one another. Each form description in the template file is associated with a regular expression and a list of controls. Each control has an index, a symbol, and a control type description.

The index associates a number with each control that resides on a form. For example, if a form contains a button control and a select box control the template file may identify information about each of these controls by asso-

ciating the number zero with the button control and the number one with the select box control. The template file may also contain a plain text name for each control. In one embodiment of the invention, each control is associated with a symbol. The symbol represents the data that is to be placed inside the fields of the form. For example, the symbol can represent a name, date, or gender. Each control also has a control type description that identifies what type of control is present. The control type description, for example, may be utilized by the form completion program to determine what steps are required to populate the control with data. The way a text box is filled, for example, may be different from the way a check box is completed. A text box may accept raw textual data whereas a check box may accept a Boolean value.

In one embodiment of the invention, each form in the template file has a regular expression associated with it. The regular expression describes a set of one or more forms that a particular template file resembles (e.g. by using a string of text, such as a URL, that is contained within a form) and thereby provides information that may be used to perform a string matching operation. String matching provides the form completion program with a way to determine whether the template file resembles a form to within a certain threshold. For example, if the URL of a form contains a word that starts with W followed by a period and three more letters, this information can be stored in the regular expression and used to verify that the correct template file is being used to complete the form.

If the form completion program attempts to complete a form that is not represented in the template file, a list of the data that was collected from the user is displayed. When the list is displayed, the user may elect to manually intervene by dragging information from the list onto the form. If, for example, the user wishes to place address information into one of the data receptacles contained on a form, the user can drag the address data from the list to the form. At this point, the information placed over the form is inserted into the appropriate data receptacle. The user may repeat this process to fill-in the remaining data receptacles. In one embodiment of the invention, the actions of the user may be used to generate a form description. For example, if a user manually completes a form, the form completion program may track the information the user elected to place in the forms various data receptacles. This information may then be transmitted to a third party for verification and utilized to generate a new form description for the template file. In one embodiment of the invention, a different template file is created for each form. However, a single template file may represent more than one form. For example, a generic template that operates on forms that adhere to a set of predefined standards may be created and later verified by an independent party. The template may store information using a variety of formats such as binary, hexadecimal, or text (e.g. ASCII or XML). It is also possible to use other methods for representing information. A template file, for example, may be an executable file, a compressed file, and/or any other form of storing digital information. For example, a template file may be included in the form (using ECML, inband tems).

The template files are typically stored on the computer hosting the target application in a template directory. The form completion program is configured to search for templates that resemble the form image in the template directory to within a certain threshold. However, the invention also contemplates searching for templates at alternate locations. Templates, for example, may reside on any other computer that is accessible via a telecommunication medium such as a computer network.

The form completion program examines each template file in order to determine if one or more of the template files resembles the form image to within a certain threshold. For example, if a percentage of the template file is similar to the form image then the form completion program may assume that a match occurred. In one embodiment of the invention, the threshold percentage is thirty percent. However, the present invention contemplates the use of any percentage indicative of a match. For example, in some instances a threshold percentage lower than thirty percent indicates a match whereas in other instances a percentage greater than thirty percent indicates a match. The percentage chosen is largely a result of how much the form has changed since the template file was created. If, for example, a form has added new graphics, but has substantially the same text as when the template was created, then the template file will resemble the form image to a greater percent than it would if the text or number of data receptacles on the form were changed.

If a template file that resembles the form image to within a certain threshold is located, then the form completion program proceeds to step **517** where it acknowledges that a match occurred. If no match is found the form completion program informs the user that the data population command was not successfully completed. At this point the program may execute step **519** and exit. Otherwise, the form completion program executes step **518** where the information obtained from the template file is utilized to populate the form.

The form completion program utilizes the template file to identify what kind of data to insert into each of the form's data receptacles. For example, the template file allows the form completion program to determine which of the data receptacles contain personal information and which data receptacles contain payment information. Once the form completion program successfully identifies what kind of data to insert into each data receptacle the program begins to input the appropriate kind of data into the appropriate data receptacle. The user's name, for example, is placed in all data receptacles that expect such information and the user's credit card number is placed in all data receptacles that expect credit card information.

The present invention contemplates the use of multiple techniques to insert information into the data receptacles. The technique used is largely dependent upon the type of target application used to display the form. If, for example, the form is displayed using the Internet Explorer web browser the form completion program populates the data receptacles by sending a command to the browser that tells the browser which string belongs in which data receptacle (e.g. insert name into data receptacle 1). If the form is displayed with the Netscape web browser, the technique for inserting information into the data receptacles differs. When Netscape is used, data is inserted by grabbing standard input and emulating keystrokes. Each keystroke inserts a character of information into the appropriate data receptacle. To move from one data receptacle to the next the set focus system message is sent (WM_SETFOCUS). However, the invention contemplates using other methods such as emulating the tab key. Once information is properly inserted into the appropriate data receptacles, the data population command is complete.

The Form Completion Program

In one embodiment of the invention, the form completion program uses a registry with the operating system (OS) to assure proper execution. The registry contains multiple fields. Each field contains information that may be utilized by the form completion program during execution. An

example of the kind of fields that may be placed in the registry follows.

Field Name	Description
InstallPath	Contains the path of installed program
UpdateFile	Contains the name and path to copy to be deleted
LastUser	Contains the name of last user
Coordinate X	Contains the horizontal coordinate (in pixel), of the program window
Coordinate Y	Contains the vertical coordinate (in pixel), of the program window
Server URL	Contains URL(s) where update routine should check for a new version of the program

When the form completion program is installed, a directory structure is created that contains files used by the form completion program during execution. This directory contains the form completion program and two sub-directories: "user" and "data". The "user" sub-directory contains as many sub-directories as there are users. Each sub-directory contains one file called by the login name of the user, followed by the extension ".dat". This file contains encrypted data about the user (e.g. address, credit card information, electronic money, etc . . . ,). The "data" sub-directory contains general information utilized by the form completion program during execution. For example, graphical images that represent various kinds of data, web site information, and other information useful to the form completion program is placed in the "data" sub-directory. The information placed in the "data" sub-directory is organized in a tree of one or more additional sub-directories.

The "domain" sub-directory, for example, is located within the "data" sub-directory. In one embodiment of the invention, the "domain" sub-directory contains a database of information that is used by the form completion program to populate a form. For example, the database may contain a list of web sites (e.g. domain names) having forms that can be populated. Template files associated with each domain name are also placed in the database.

The database contains sub-directories listing each of the top-level domain names (e.g..com, .net,. org, etc . . .). Each sub-directory of top-level domain names contains a tree of sub-directories organized in alphabetical order. When the form completion program initially launches, a startup routine begins to execute. The startup routine performs a number of checks to ensure that the program is running properly. The form completion program then launches a separate process, called a thread, to determine if the form completion program that is running is the most current version available. The process determines if the program is current by contacting a server. The server responds by transmitting an update signal to the form completion program that indicates whether or not the program is indeed out of date. If the form completion program is out of date, the program starts the download-update process.

After running the start-up checks the form completion program enters a listening mode. While in listening mode the program listens for any action that occurs within the display space of the form completion program, or hot keys. For example, the program listens for actions received from the operating system (e.g. mouse pointer selection, keyboard key stroke, voice commands, etc . . .). These actions are handled by the application action listener which generates commands according to an action-to-command map.

FIG. 6 illustrates some of the actions the form completion program listens for when it is in the listening mode. When an action is triggered that is bound to a command (e.g. those being listened for), the form completion program runs the segment of code responsible for performing the corresponding action. If, for example, the form completion program is in listening mode **600** and the user initiates a login action **601**, then the program performs a login operation **602**. In one embodiment of the invention, the login operation ensures that users are properly authenticated before being allowed access to sensitive data. For example, users are not permitted to access personal information such as a credit card number without first being authenticated. A user is authenticated when the user enters the correct login and password information.

However, in one embodiment of the invention, the user's password information is not stored by the form completion program. To determine if the user's login and password information is correct, the form completion program determines if the user has logged in before by examining whether a directory having the name entered by the user as a login parameter exists. If the user has not previously logged in, the user is asked to provide a login name and a password. If the user has previously logged in (e.g. the user's directory exists), then the program generates a 32-bit number using a CRC Algorithm. A random number key is generated using the CRC output as a seed. If the correct random number is generated, the user is authenticated. At this point, the user is permitted to access the user's sensitive data. This technique is beneficial because it enables the form completion program to authenticate a user without having to permanently store the user's password information. The present invention also contemplates the use of various authentication schemes that are known in the art and is not limited to the process discussed herein.

In one embodiment of the invention, after the login operation is performed the form completion program proceeds to step **603** where it determines if the user is a new user. If the user is new, the form completion program advances to step **604** where it shows the user a data collection interface. Once the user enters data into the data collection interface, the program executes step **605** where it displays a shopping view to the user. The shopping view is comprised of one or more graphical representations of the user's sensitive information.

It is possible for other types of actions to transpire while the form completion program is in listening mode. For example, the user might wish to perform a shopping action **606**. When a shopping action **606** occurs the form completion program performs a shopping operation **607**. In one embodiment of the invention, a shopping action occurs when the user selects an image indicative of a particular shopping destination. For example, if the form completion program displays a list of web sites to the user, a shopping action **606** occurs when the user selects one of the web sites on the list. When the user initiates the shopping action **606** it causes the form completion program to perform step **607** where a web browser window is displayed to the user containing the web page the user selected from the list.

When a search action **608** occurs, the form completion program performs a search **609**. For example, in one embodiment of the invention the form completion program contains a location for the user to enter a search query and a way to identify a preferred search engine. If the user enters a search query and submits the query a search **609** is performed. The form completion program responds by passing the search parameter entered by the user to the preferred search engine identified by the user. The results of the search are displayed inside of a web browser window that runs external to the form completion program.

If an update action **610** occurs, the form completion program performs an update **611**. During an update operation **611** the form completion program checks to see if the version of the program currently executing is the most recent version available. In one embodiment of the invention, the update operation **611** queries a server via a network to determine the latest available version is. If the form completion program currently executing is in need of a version update, the program starts an update thread to handle the update process. When the execution of the update process is complete, the form completion program running is the most recent version. The update operation **611**, for example, may be utilized to add functionality to the form completion program.

The form completion program may be configured to listen for actions other than the ones identified in FIG. **6**. For example, the user may customize the program to listen for a specific type of action. The present invention also contemplates action added by the update operation **611**.

When a FillOut Information action **612** is performed the form completion program executes step **613**. At step **613**, the user is prompted for information. In one embodiment of the invention, the user is shown a data collection interface that provides locations for the user to enter a particular type of information. For example, the user may be prompted for personal information, payment information, and/or any other type of information deemed necessary to populate a form. This information may be associated with an image file, text file, multimedia file, or other representation of the data entered into the data collection interface.

If a FillOut Form action **614** occurs, the form completion program executes a data population command **615**. The data population command **615** provides the user with a mechanism for populating a form. When the user elects to stop running the form completion program, a quit action **616** occurs. When a quit action **616** executes the form completion program returns to an inactive state.

Referring now to FIG. **7**, an example of several of the interfaces presented to the user by the form completion program is shown. When the user initially opens the form completion program, a login interface **715** is displayed which provides the user with locations for a user name **711** and password **712**. In one embodiment of the invention, a login action **601** occurs when the user enters the appropriate information into the login interface **715** and presses the return key.

If the user is properly authenticated the form completion program displays a data collection interface **720**. the data collection interface **720** contains one or more fields **705–709** for entering data. When the user enters data into these fields **705–709** the data is stored for later use during a form population command. There are several different types of data collection interfaces **720**. Each type is used to obtain a different data set from the user. For example, the user is presented one interface for entering payment information and another interface for entering personal information.

If, for example, the data collection interface **720** intended for collecting payment information is displayed, the user is provided with one or more fields **705–709** for entering data. Each field is intended to store a different type of information. For example, field **705** is used to collect the form of payment (e.g. American Express) the user wishes to utilize for later populating a form. Fields **706–709** provides a place for the user to enter information related to the form of payment identified. The name of the issuing bank is entered into field **706**. The account number of the identified card is entered into field **707** and the expiration date and name on the card are entered into fields **708** and **709**.

Once the user enters such information into fields **705–709** it is stored by the form completion program for later use. In one embodiment of the invention, the data collected via the data collection interface is stored on the computer having the form completion program in encrypted form and protected from unauthorized users by an authentication mechanism. For example, data about a particular user may be stored in a directory bearing the user's login name. Before access is granted to the information in that directory the user is prompted for a name and password. If the appropriate user name and password is entered, then access is granted to the specified user's data. When the correct information is not entered access is denied. This permits multiple users to make use of the form completion program and prevents unauthorized users from inappropriately obtaining access to another users data.

In one embodiment of the invention, each collection of data entered into the data collection interface **720** is associated with a representation of that data. The representation may be in the form of image data, text data, multimedia data, and/or any other kind of data. For example, if data collection interface **720** prompts the user for payment information, a graphic representing that payment information (e.g. a Visa Card graphic) will be displayed to the user.

For example, the shopping interface **730** displays numerous representations **701–702** of each data set entered by the user via the data collection interface **720**. Each representation **701–702** may represent a different data set. For example, representation **701** might represent a form of payment whereas representation **702** might represent a user's personal information.

Additional representations may be displayed when the user scrolls either up or down using command buttons **731** and **732**.

In one embodiment of the invention, a portion of the shopping interface **730** is dedicated to displaying a list **710** of web sites where the user can purchase goods and/or services. This list **710** can be changed and modified by the update action **610**. The list **710** may be organized in any manner preferred by the user. In one embodiment of the invention, the list **710** is divided into categories. When the user selects a link to a particular web site from list **710**, the form completion program passes the URL information identified by the users action to a web browser program. The browser program, for example, accesses the web site identified in the URL to obtain the web page identified in the URL. If after perusing through the web site the user needs to complete a form to purchase an item, the user may use the form completion program to populate the form.

To illustrate, when the user encounters a form (e.g. after shopping), the user may elect to populate the data receptacles **751–755** of that form with the information associated with a particular representative graphic. If, for example, the user wishes to fill-out form **750** with payment information, the user may cause a data population command to execute by utilizing a pointing device to drag representation **701** along path **725** until the pointing device resides over form **750**. When the pointing device is located over form **750** the data population command executes. The data population command causes data receptacles **751–755** to fill with data.

In one embodiment of the invention, the user is not required to drag representation **701** over to form **750**, but can instead drag an iconosized version of representation **701** to form **750** in order to initiate a data population command. The iconosized version of representation **701** is a smaller image of graphical representation **701** that requires less processor overhead to rapidly move about the display region. The

invention also contemplates the use of other mechanisms to initiate a data population command. The form completion program can, for example, be configured to perform a data population command whenever the user performs an event that discernibly indicates the user wishes to populate a form (e.g. a click event or a menu event).

Embodiment of Computer Execution Environment (Hardware)

An embodiment of the invention can be implemented as computer software in the form of computer readable code executed on a general purpose computer such as computer **800** illustrated in FIG. **8**, or in the form of bytecode class files executable within a Java™ runtime environment running on such a computer, or in the form of bytecodes running on a processor (or devices enabled to process bytecodes) existing in a distributed environment (e.g., one or more processors on a network). A keyboard **810** and mouse **811** are coupled to a system bus **818**. The keyboard and mouse are for introducing user input to the computer system and communicating that user input to processor **813**. Other suitable input devices may be used in addition to, or in place of, the mouse **811** and keyboard **810**. I/O (input/output) unit **819** coupled to system bus **818** represents such I/O elements as a printer, A/V (audio/video) I/O, etc.

Computer **800** includes a video memory **814**, main memory **815** and mass storage **814**, all coupled to system bus **818** along with keyboard **810**, mouse **811** and processor **813**. The mass storage **814** may include both fixed and removable media, such as magnetic, optical or magnetic optical storage systems or any other available mass storage technology. Bus **818** may contain, for example, thirty-two address lines for addressing video memory **814** or main memory **815**. The system bus **818** also includes, for example, a 64-bit data bus for transferring data between and among the components, such as processor **813**, main memory **815**, video memory **814** and mass storage **814**. Alternatively, multiplex data/address lines may be used instead of separate data and address lines.

In one embodiment of the invention, the processor **813** is a microprocessor manufactured by Motorola, such as the 680X0 processor, or a microprocessor manufactured by Intel, such as the 80X86, or Pentium processor. However, any other suitable microprocessor or microcomputer may be utilized. Main memory **815** is comprised of dynamic random access memory (DRAM). Video memory **814** is a dual-ported video random access memory. One port of the video memory **814** is coupled to video amplifier **816**. The video amplifier **816** is used to drive the cathode ray tube (CRT) raster monitor **817**. Video amplifier **816** is well known in the art and may be implemented by any suitable apparatus. This circuitry converts pixel data stored in video memory **814** to a raster signal suitable for use by monitor **817**. Monitor **817** is a type of monitor suitable for displaying graphic images.

Computer **800** may also include a communication interface **840** coupled to bus **818**. Communication interface **840** provides a two-way data communication coupling via a network link **841** to a local network **844**. For example, if communication interface **840** is an integrated services digital network (ISDN) card or a modem, communication interface **840** provides a data communication connection to the corresponding type of telephone line, which comprises part of network link **841**. If communication interface **840** is a local area network (LAN) card, communication interface **840** provides a data communication connection via network link **841** to a compatible LAN. Wireless links are also possible. In any such implementation, communication interface **840** sends and receives electrical, electromagnetic or optical signals which carry digital data streams representing various types of information.

Network link **841** typically provides data communication through one or more networks to other data devices. For example, network link **841** may provide a connection through local network **844** to local server computer **843** or to data equipment operated by an Internet Service Provider (ISP) **844**. ISP **844** in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" **845**. Local network **844** and Internet **845** both use electrical, electromagnetic or optical signals which carry digital data streams. The signals through the various networks and the signals on network link **841** and through communication interface **840**, which carry the digital data to and from computer **800**, are exemplary forms of carrier waves transporting the information.

Computer **800** can send messages and receive data, including program code, through the network(s), network link **841**, and communication interface **840**. In the Internet example, remote server computer **846** might transmit a requested code for an application program through Internet **845**, ISP **844**, local network **844** and communication interface **840**.

The received code may be executed by processor **813** as it is received, and/or stored in mass storage **814**, or other non-volatile storage for later execution. In this manner, computer **800** may obtain application code in the form of a carrier wave.

Application code may be embodied in any form of computer program product. A computer program product comprises a medium configured to store or transport computer readable code, or in which computer readable code may be embedded. Some examples of computer program products are CD-ROM disks, ROM cards, floppy disks, magnetic tapes, computer hard drives, servers on a network, and carrier waves.

The computer systems described above are for purposes of example only. An embodiment of the invention may be implemented in any type of computer system or programming or processing environment.

Thus, a method and apparatus for populating forms is described.

What is claimed is:

1. A method for populating a form with data comprising:

obtaining a form from a source location; said form having at least one data receptacle;

displaying said form to a user;

obtaining data for populating said form;

storing said data in a secure storage medium;

obtaining values for said data receptacles of said form from said secure storage medium when said form resembles a portion of a template file to a certain threshold.

2. The method of claim **1** further comprising:

determining when said form resembles a template file by examining a regular expression.

3. The method of claim **1** further comprising:

associating a graphical representation with said data.

4. The method of claim **1** wherein said data is obtained from said user.

5. The method of claim **1** wherein said data is obtained from a data provider via a network communication medium.

6. The method of claim **1** further comprising:

authenticating said user before access to said secure storage medium is permitted.

7. The method of claim 1 wherein said data stored in said secure storage medium is encrypted.

8. The method of claim 1 wherein said secure storage medium resides on a web client.

9. The method of claim 2 wherein said template file comprises at least one form description.

10. The method of claim 9 wherein said form description is associated with said regular expression.

11. The method of claim 9 wherein said form description is associated with a list of controls comprising one or more controls.

12. The method of claim 11 wherein each of said one or more controls is associated with an index, a symbol, and a control type description.

13. A method for populating a form with data comprising:

displaying a form having at least one data receptacle;

displaying a form completion interface having a portion representative of a data set;

determining if said form resembles a template file to a given threshold level;

populating said at least one data receptacle with said data set when said portion is moved from said form completion interface to said form.

14. The method of claim 13 further comprising:

presenting a first display region to a user;

obtaining said data set from said user via said first display region;

granting said user access to a second display region having a graphical representation associated with said data set when said user is authenticated.

15. The method of claim 13 further comprising:

storing said data set in a secure storage medium.

16. The method of claim 13 wherein said step of determining if said form resembles said template file to within a certain threshold is accomplished by comparing a form image of said form to said template file.

17. The method of claim 13 further comprising obtaining said template file from a web client.

18. The method of claim 13 further comprising obtaining said template file from a server computer.

19. A computer program product comprising:

a computer usable medium having computer readable program code embodied therein for populating a form, said computer program product comprising:

computer readable program code configured to cause a computer to obtain a form from a source location, said form having at least one data receptacle;

computer readable program code configured to cause a computer to display said form to a user;

computer readable program code configured to cause a computer to obtain data for populating said form;

computer readable program code configured to cause a computer to store said data in a secure storage medium;

computer readable program code configured to cause a computer to obtain values for said data receptacles of said form from said secure storage medium when said form resembles a template file to within a certain threshold.

20. The computer program product of claim 19 further comprising computer readable program code configured to cause a computer to obtain data from a user.

21. The computer program product of claim 19 further comprising computer readable program code configured to cause a computer to obtain data from a data provider.

22. The computer program product of claim 19 further comprising a computer readable program code configured to cause a computer to associate a graphical representation with said data.

23. The computer program product of claim 19 further comprising computer readable program code configured to cause a computer to obtain a form image representative of said form.

24. The computer program product of claim 19 further comprising computer readable program code configured to cause a computer to compare said form image to said template file and determine if said form image resembles said template file.

25. The computer program product of claim 19 further comprising computer readable program code configured to authenticate said user before access to said secure storage medium is permitted.

26. The computer program product of claim 19 wherein said data in said secure storage medium is encrypted.

27. The computer program product of claim 19 wherein said secure storage medium resides on a web client.

28. A system for populating a form with data comprising:

a display region configured to display a form having at least one data receptacle; and

a form completion program configured to copy data into said at least one field when a form resembles at least one of a plurality of template files.

29. The system of claim 28 wherein said data is stored in a secure storage medium.

30. The system of claim 29 wherein said secure storage medium resides on a web client.

31. The system of claim 29 wherein said secure storage medium resides on a server computer.

32. The system of claim 28 wherein said form completion program collects said data from a user via a data collection interface.

33. A system for supplying data to a form comprising:

a computational device configured to display a form having at least one data receptacle, said computational device configured to perform the steps of:

obtaining data about a user; and

populating said at least one data receptacle with said data when said form resembles at least one of a plurality of template files.

* * * * *

US005870717A

United States Patent [19]

Wiecha

[11] Patent Number: 5,870,717

[45] Date of Patent: Feb. 9, 1999

[54] **SYSTEM FOR ORDERING ITEMS OVER COMPUTER NETWORK USING AN ELECTRONIC CATALOG**

[75] Inventor: **Charles Francis Wiecha**, New York, N.Y.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **558,065**

[22] Filed: **Nov. 13, 1995**

[51] **Int. Cl.**[6] **G06F 153/00**

[52] **U.S. Cl.** **705/26**; 235/385

[58] **Field of Search** 395/201, 226–228, 395/244; 379/91.01; 340/825.26–825.28, 825.33–825.35; 283/56; 235/378–381, 385; 902/22, 30–33; 705/1, 26–28; 707/1, 10, 200

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,992,940	2/1991	Dworkin	235/383
5,315,504	5/1994	Lemble	395/650
5,319,542	6/1994	King, Jr. et al.	235/383
5,570,291	10/1996	Dudle et al.	364/188
5,576,951	11/1996	Lockwood	395/227

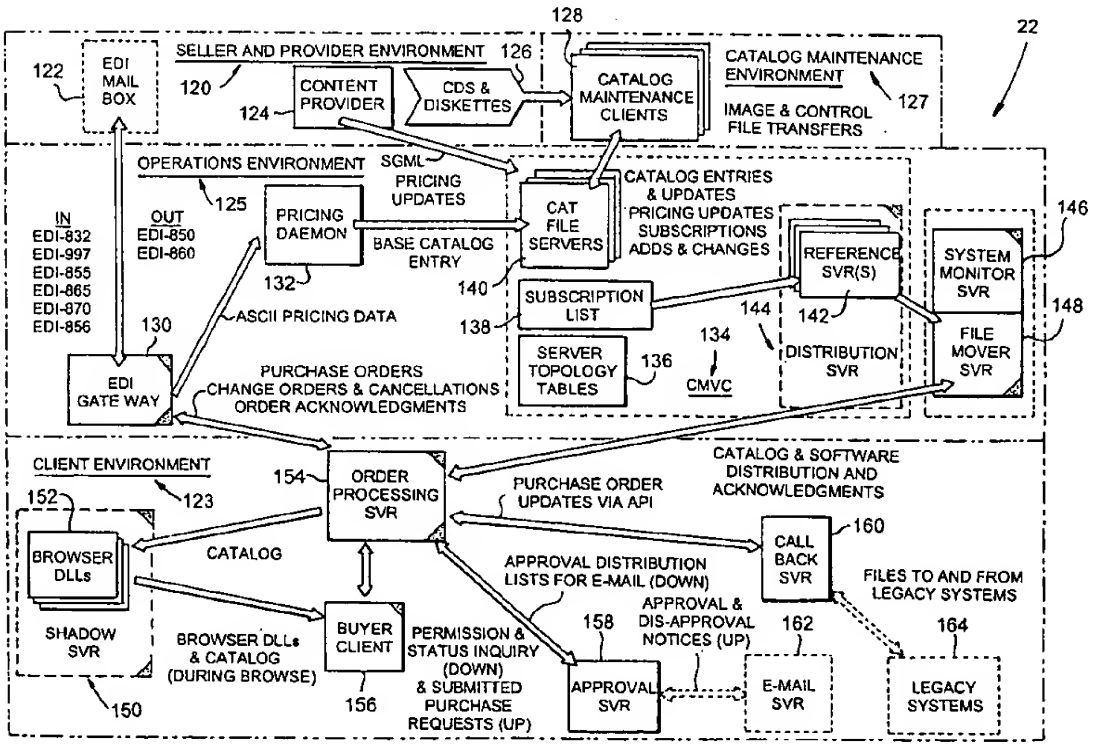
Primary Examiner—Frantzy Poinvil

Attorney, Agent, or Firm—Stephen C. Kaufman, Esq. IBM Corporation; Scully, Scott, Murphy & Presser

[57] **ABSTRACT**

Current corporate purchasing procedures are labor-intensive and therefore costly. The system enables an employee who needs an item which must be ordered from a supplier to select the item from an electronic catalog displayed on a personal computer and submit an order for approval and processing directly, by-passing both the normal paper approvals and the manual verification of the order by the organization's Purchasing department. It achieves this by means of an electronic catalog accessible from the employee's own personal computer, and a computer network and associated services linking the enterprise to one or more suppliers.

6 Claims, 12 Drawing Sheets



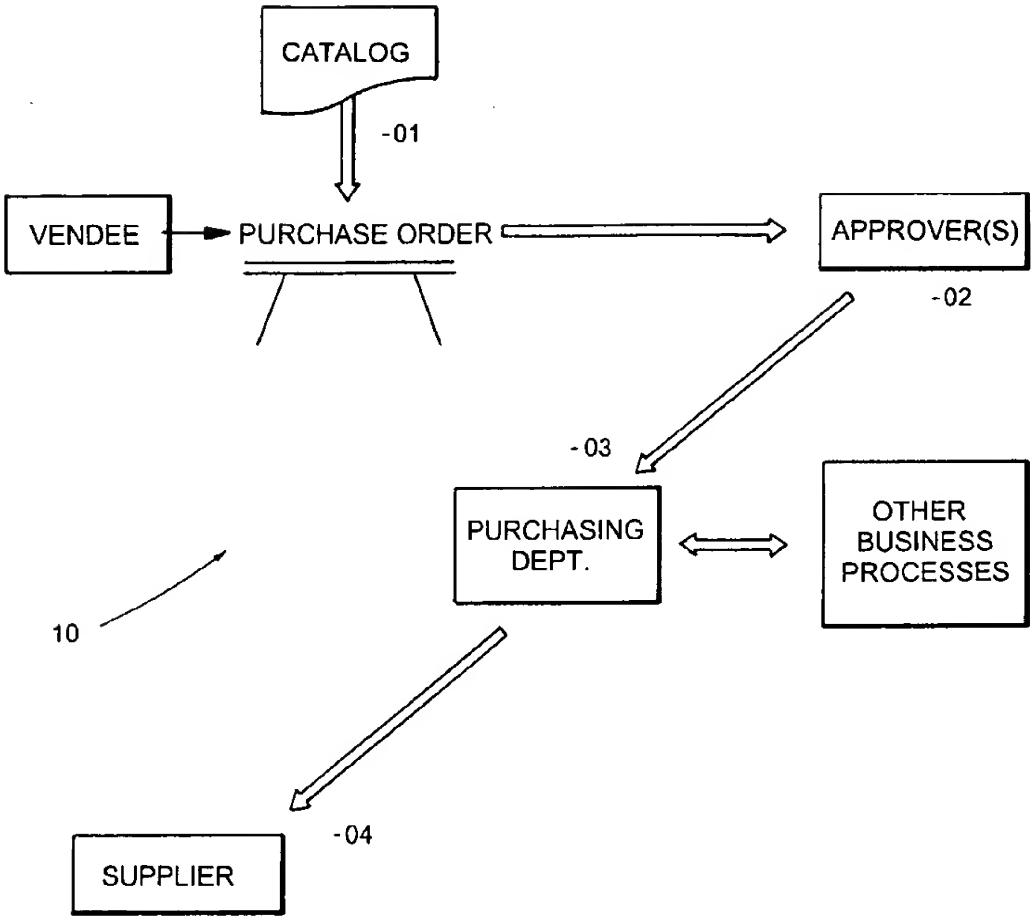


Fig. 1
(Prior Art)

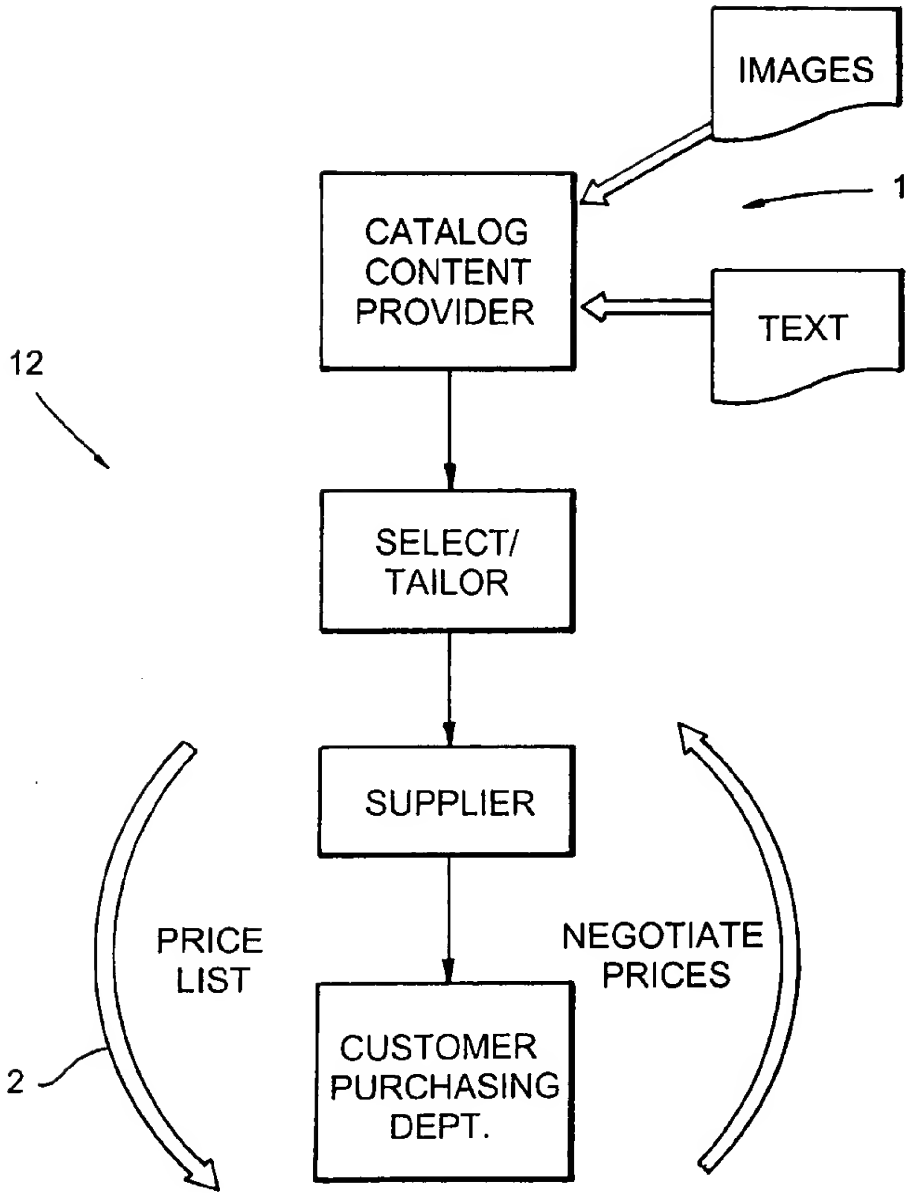


Fig. 2
(Prior Art)

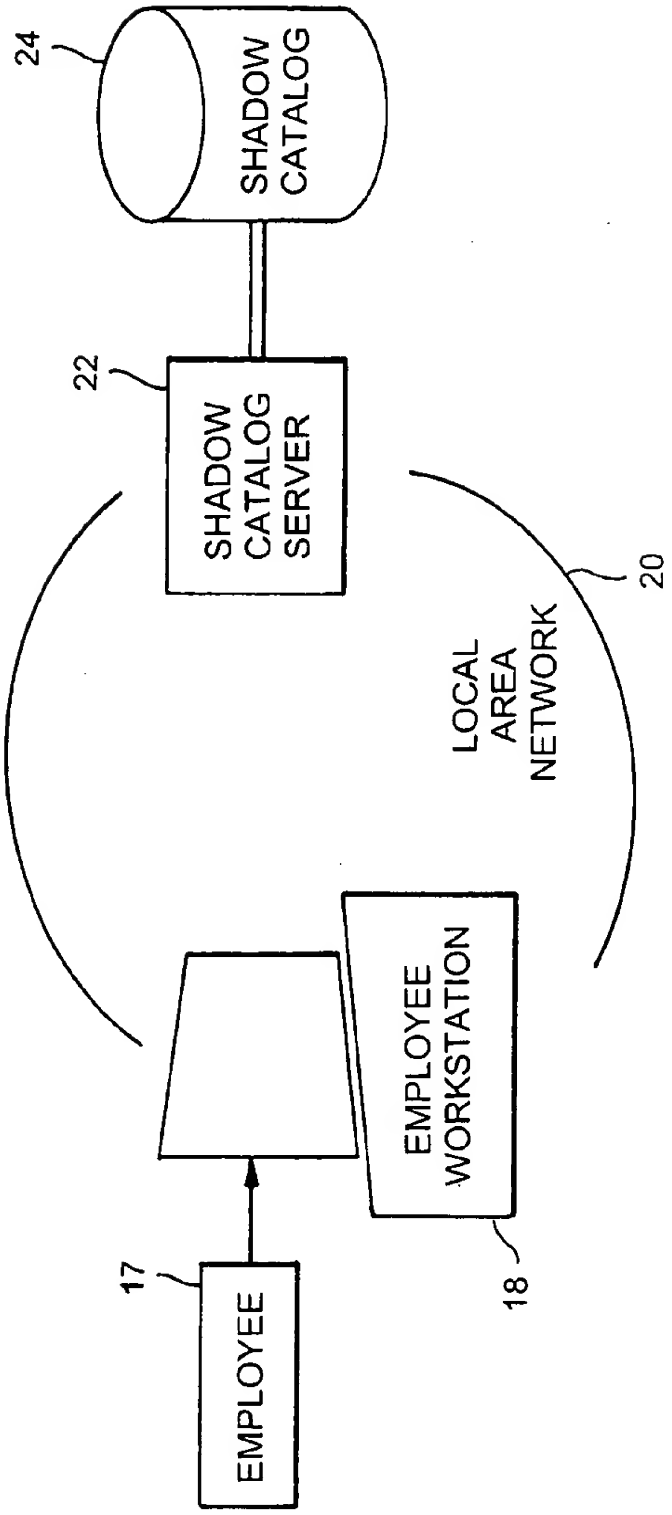

24
SHADOW CATALOG
22
SHADOW CATALOG SERVER
20
LOCAL AREA NETWORK
EMPLOYEE WORKSTATION
18
14
17
EMPLOYEE

Fig. 3

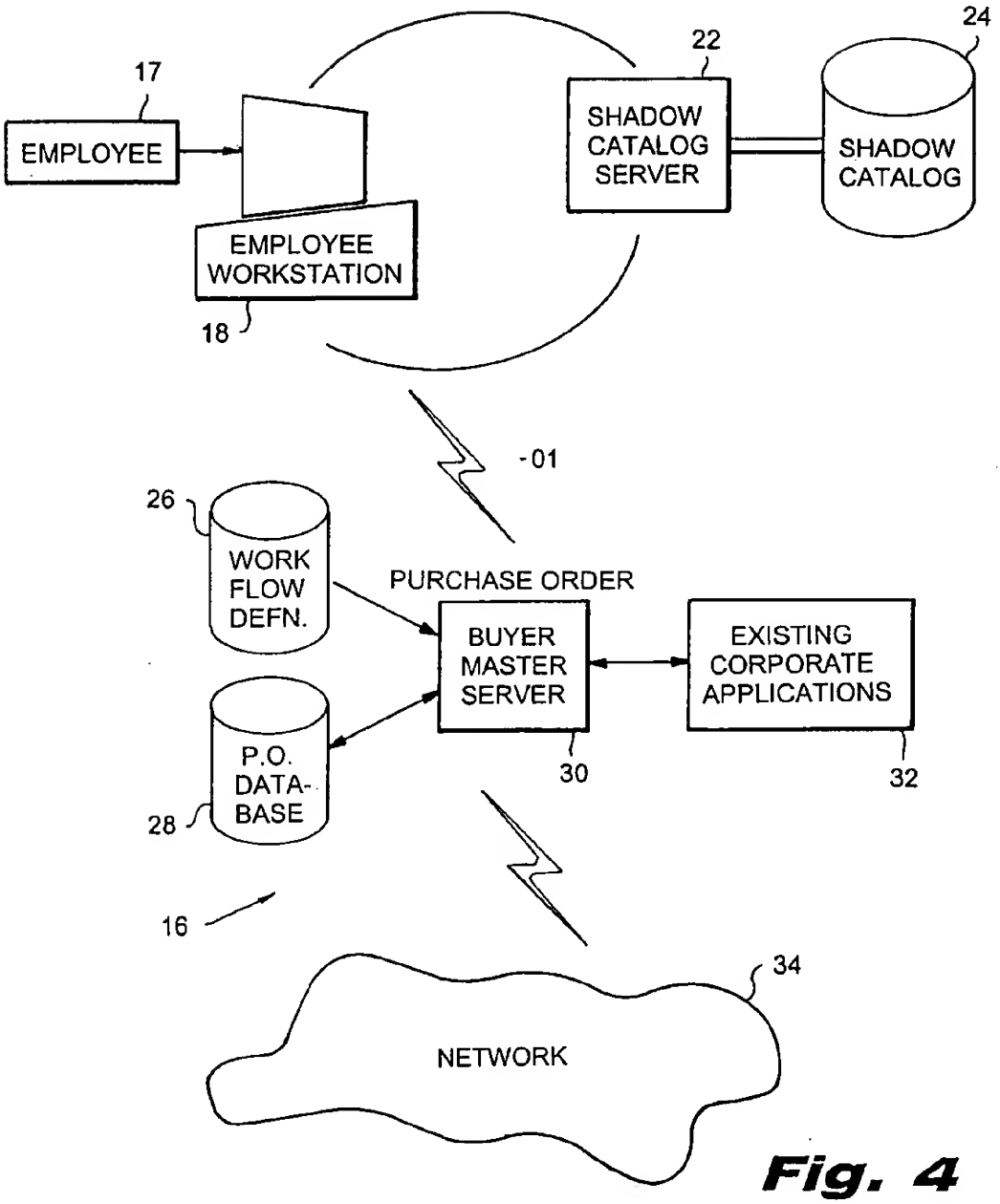

17
EMPLOYEE
EMPLOYEE WORKSTATION
18
22
SHADOW CATALOG SERVER
24
SHADOW CATALOG
-01
26
WORK FLOW DEFN.
PURCHASE ORDER
BUYER MASTER SERVER
30
EXISTING CORPORATE APPLICATIONS
32
P.O. DATA-BASE
28
16
34
NETWORK

Fig. 4

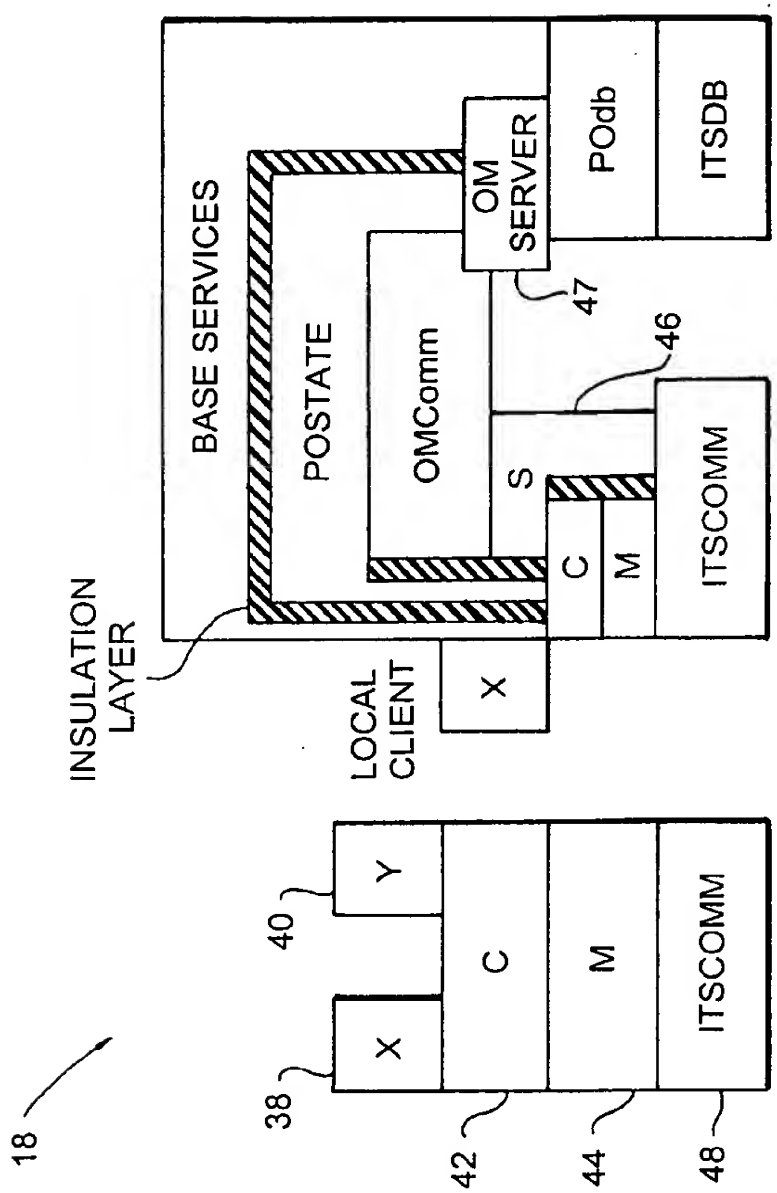

18
INSULATION LAYER
BASE SERVICES
POSTATE
OMComm
OM SERVER
POdb
ITSDB
47
46
S
C
M
ITSCOMM
LOCAL CLIENT
X
40
Y
38
X
C
M
ITSCOMM
42
44
48

Fig. 5

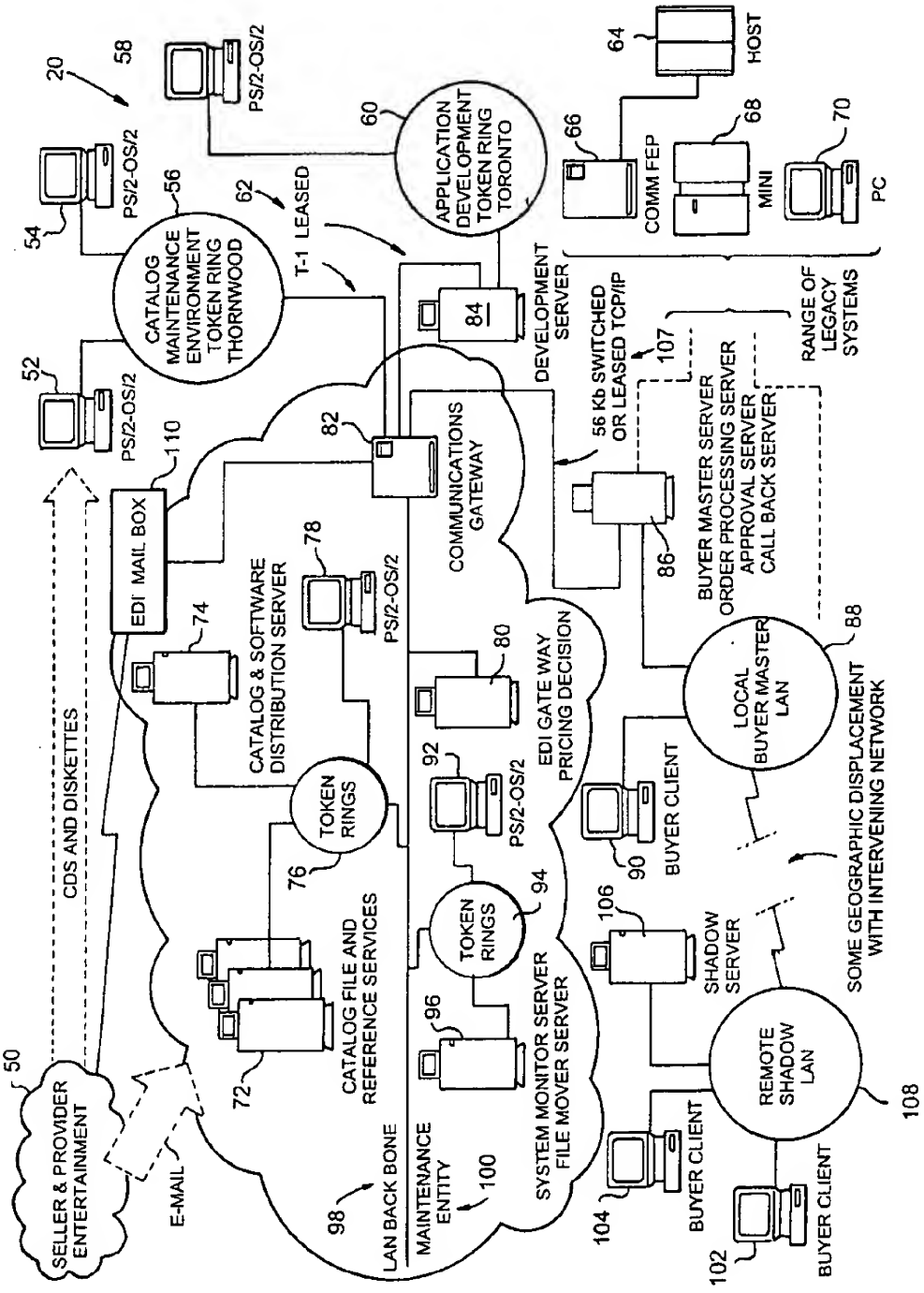

SELLER & PROVIDER ENTERTAINMENT
CDS AND DISKETTES
E-MAIL
LAN BACK BONE
MAINTENANCE ENTITY
CATALOG MAINTENANCE ENVIRONMENT TOKEN RING THORNWOOD
APPLICATION DEVELOPMENT TOKEN RING TORONTO
T-1 LEASED
56 Kb SWITCHED OR LEASED TCP/IP
COMMUNICATIONS GATEWAY
EDI MAIL BOX
CATALOG & SOFTWARE DISTRIBUTION SERVER
CATALOG FILE AND REFERENCE SERVICES
TOKEN RINGS
EDI GATE WAY PRICING DECISION
SYSTEM MONITOR SERVER FILE MOVER SERVER
DEVELOPMENT SERVER
BUYER MASTER SERVER ORDER PROCESSING SERVER APPROVAL SERVER CALL BACK SERVER
RANGE OF LEGACY SYSTEMS
COMM FEP
MINI
PC
HOST
LOCAL BUYER MASTER LAN
BUYER CLIENT
SHADOW SERVER
REMOTE SHADOW LAN
SOME GEOGRAPHIC DISPLACEMENT WITH INTERVENING NETWORK
PS/2-OS/2

Fig. 6

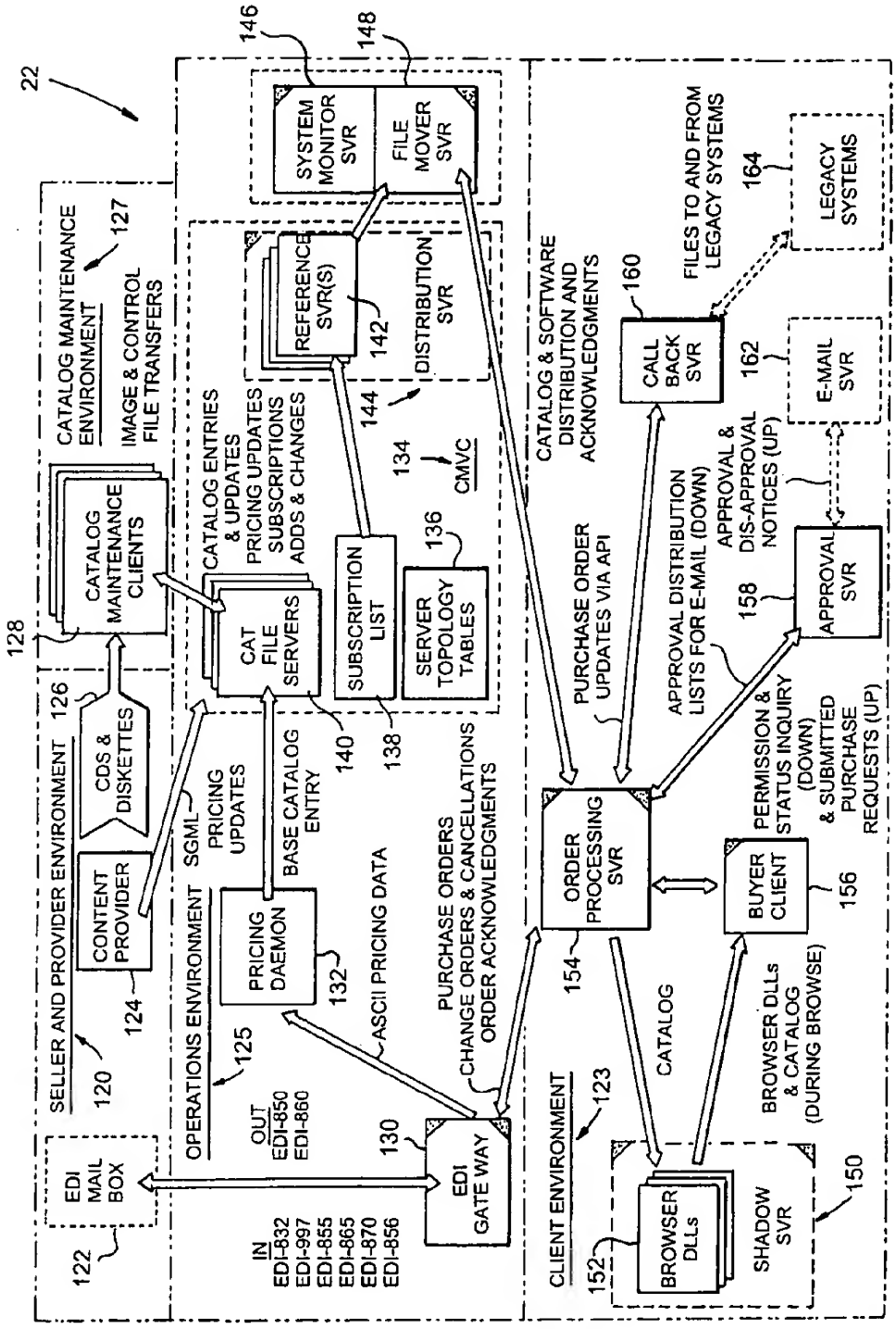

22
146
148
SYSTEM MONITOR SVR
FILE MOVER SVR
CATALOG MAINTENANCE ENVIRONMENT
127
IMAGE & CONTROL FILE TRANSFERS
REFERENCE SVR(S)
142
DISTRIBUTION SVR
144
CATALOG ENTRIES & UPDATES
PRICING UPDATES
SUBSCRIPTIONS ADDS & CHANGES
CMVC
134
CATALOG MAINTENANCE CLIENTS
128
126
CDS & DISKETTES
CAT FILE SERVERS
SUBSCRIPTION LIST
136
SERVER TOPOLOGY TABLES
CATALOG & SOFTWARE DISTRIBUTION AND ACKNOWLEDGMENTS
160
CALL BACK SVR
FILES TO AND FROM LEGACY SYSTEMS
164
LEGACY SYSTEMS
162
E-MAIL SVR
APPROVAL & DIS-APPROVAL NOTICES (UP)
158
APPROVAL SVR
APPROVAL DISTRIBUTION LISTS FOR E-MAIL (DOWN)
PURCHASE ORDER UPDATES VIA API
SELLER AND PROVIDER ENVIRONMENT
120
CONTENT PROVIDER
124
SGML
PRICING UPDATES
BASE CATALOG ENTRY
140
138
OPERATIONS ENVIRONMENT
125
PRICING DAEMON
132
ASCII PRICING DATA
PURCHASE ORDERS
CHANGE ORDERS & CANCELLATIONS
ORDER ACKNOWLEDGMENTS
ORDER PROCESSING SVR
154
PERMISSION & STATUS INQUIRY (DOWN)
& SUBMITTED PURCHASE REQUESTS (UP)
BUYER CLIENT
156
CATALOG
BROWSER DLLs & CATALOG (DURING BROWSE)
EDI MAIL BOX
122
OUT
EDI-850
EDI-860
130
EDI GATE WAY
IN
EDI-832
EDI-997
EDI-855
EDI-865
EDI-870
EDI-856
CLIENT ENVIRONMENT
123
BROWSER DLLs
152
SHADOW SVR
150

Fig. 7

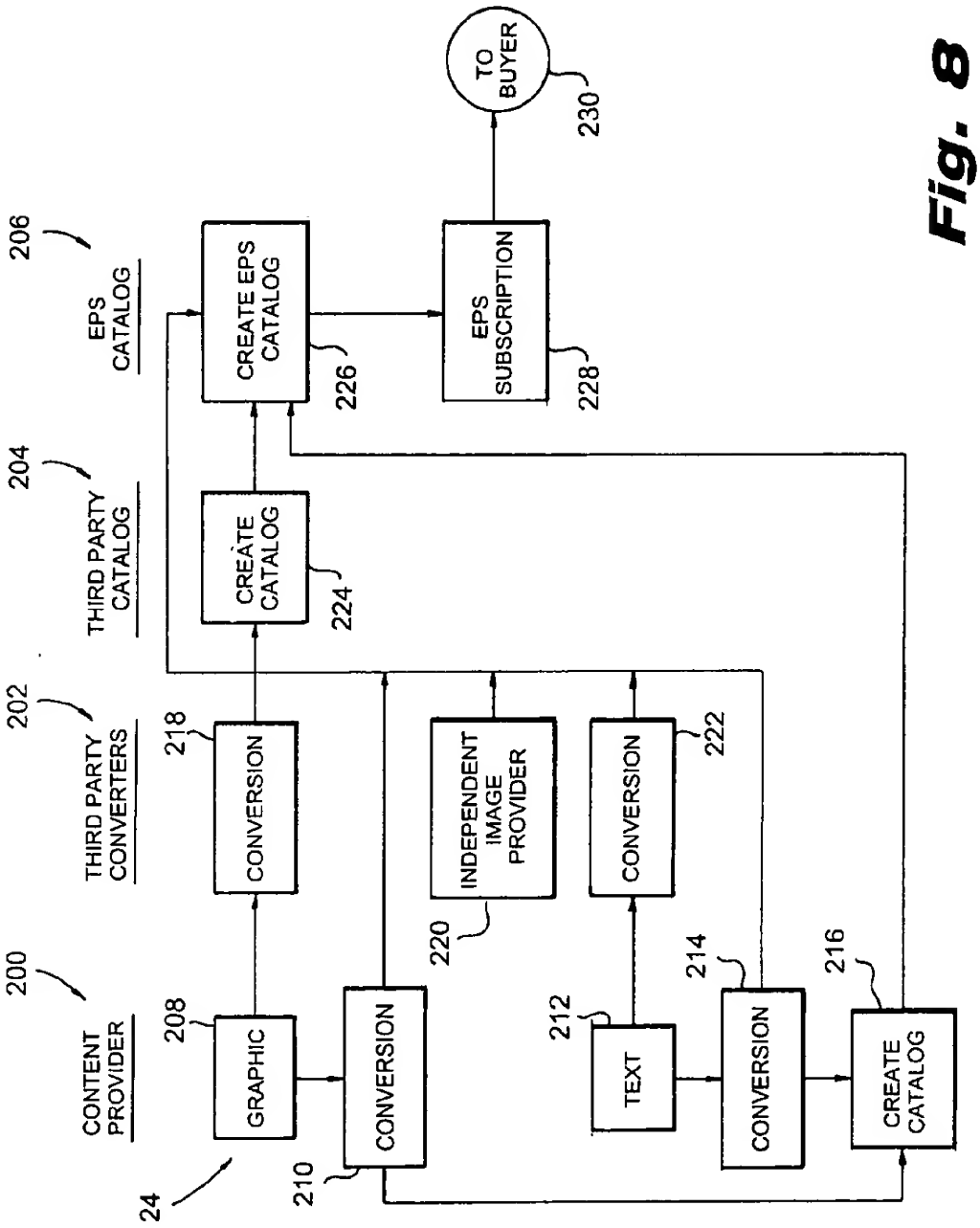

CONTENT PROVIDER
200
THIRD PARTY CONVERTERS
202
THIRD PARTY CATALOG
204
EPS CATALOG
206
24
GRAPHIC
208
CONVERSION
210
CONVERSION
218
CREATE CATALOG
224
CREATE EPS CATALOG
226
EPS SUBSCRIPTION
228
TO BUYER
230
INDEPENDENT IMAGE PROVIDER
220
TEXT
212
CONVERSION
214
CONVERSION
222
CREATE CATALOG
216

Fig. 8

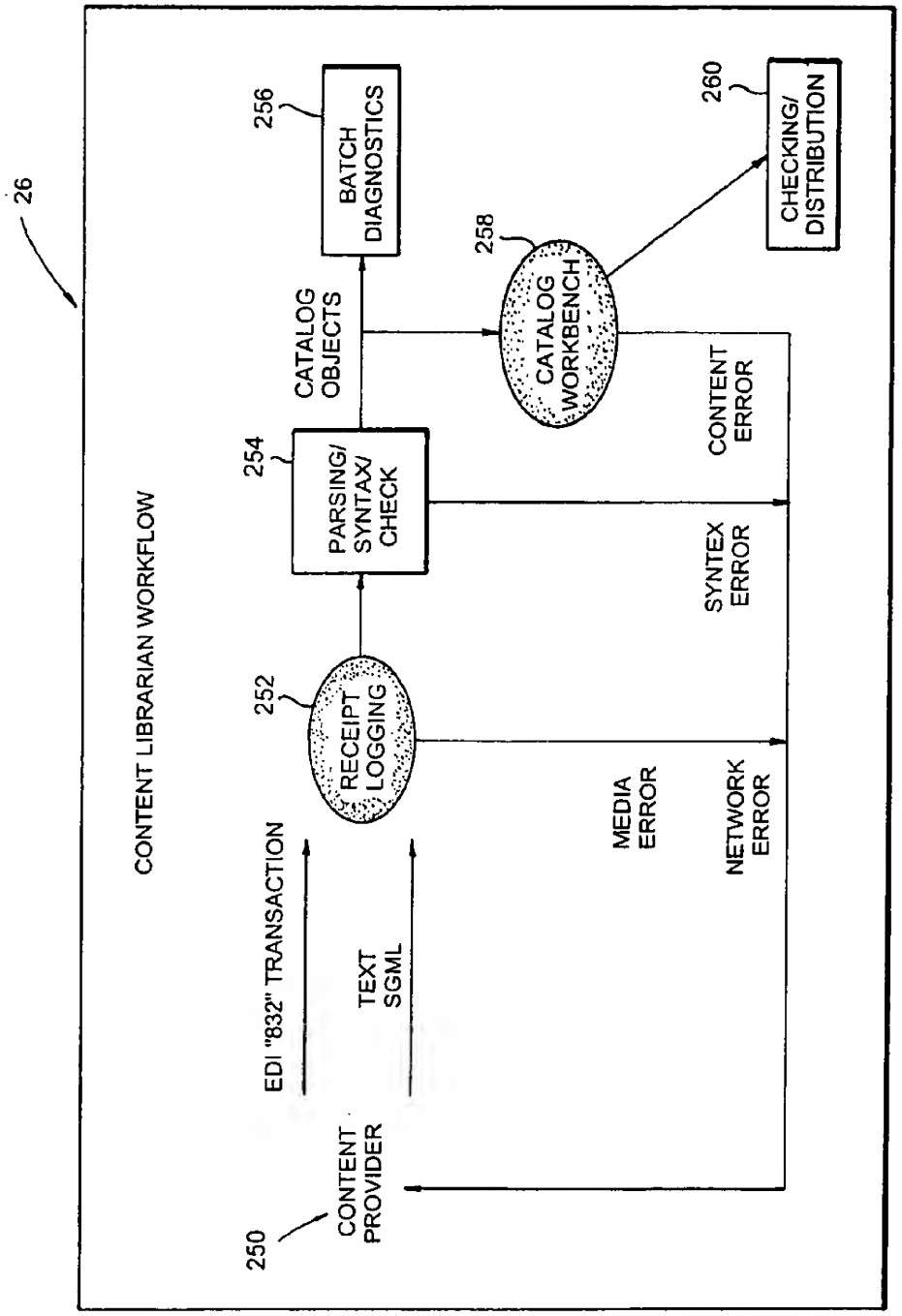

26
CONTENT LIBRARIAN WORKFLOW
250
CONTENT PROVIDER
EDI "832" TRANSACTION
TEXT SGML
252
RECEIPT LOGGING
254
PARSING/ SYNTAX/ CHECK
256
BATCH DIAGNOSTICS
CATALOG OBJECTS
258
CATALOG WORKBENCH
260
CHECKING/ DISTRIBUTION
MEDIA ERROR
NETWORK ERROR
SYNTEX ERROR
CONTENT ERROR

Fig. 9

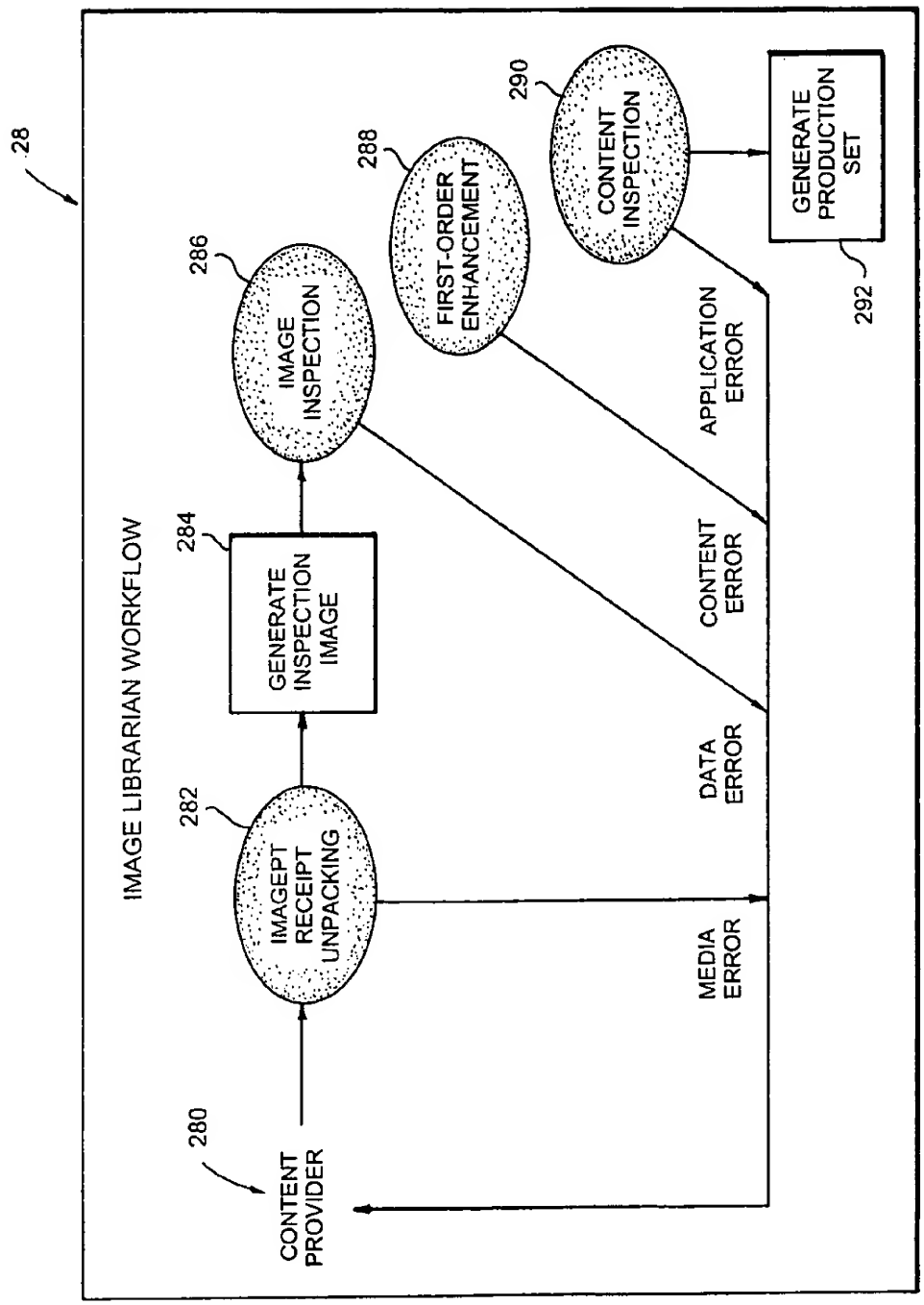

28
IMAGE LIBRARIAN WORKFLOW
280
CONTENT PROVIDER
282
IMAGEPT RECEIPT UNPACKING
284
GENERATE INSPECTION IMAGE
286
IMAGE INSPECTION
288
FIRST-ORDER ENHANCEMENT
290
CONTENT INSPECTION
292
GENERATE PRODUCTION SET
MEDIA ERROR
DATA ERROR
CONTENT ERROR
APPLICATION ERROR

Fig. 10

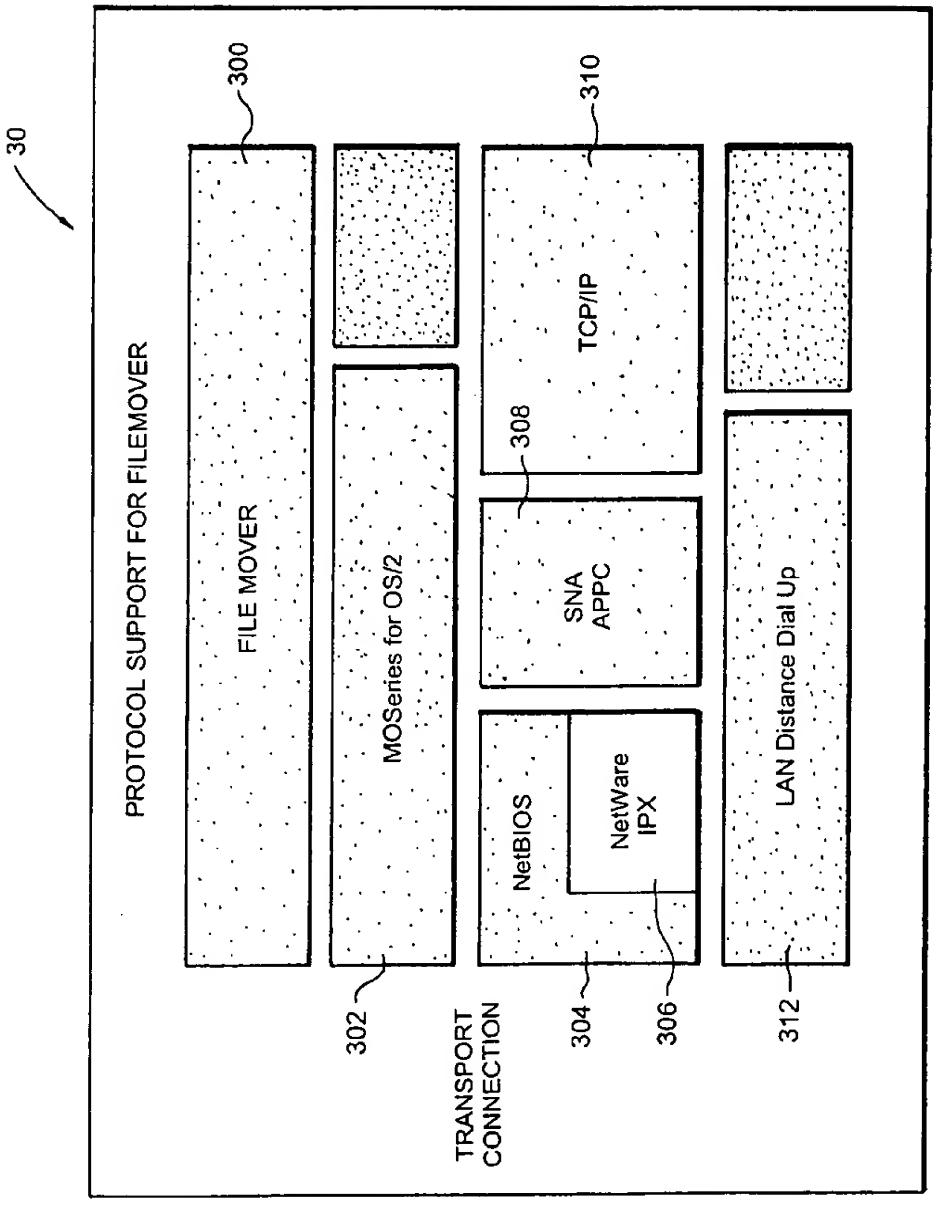

30
PROTOCOL SUPPORT FOR FILEMOVER
300
FILE MOVER
MOSeries for OS/2
302
TRANSPORT CONNECTION
NetBIOS
NetWare IPX
304
306
310
TCP/IP
308
SNA APPC
LAN Distance Dial Up
312

Fig. 11

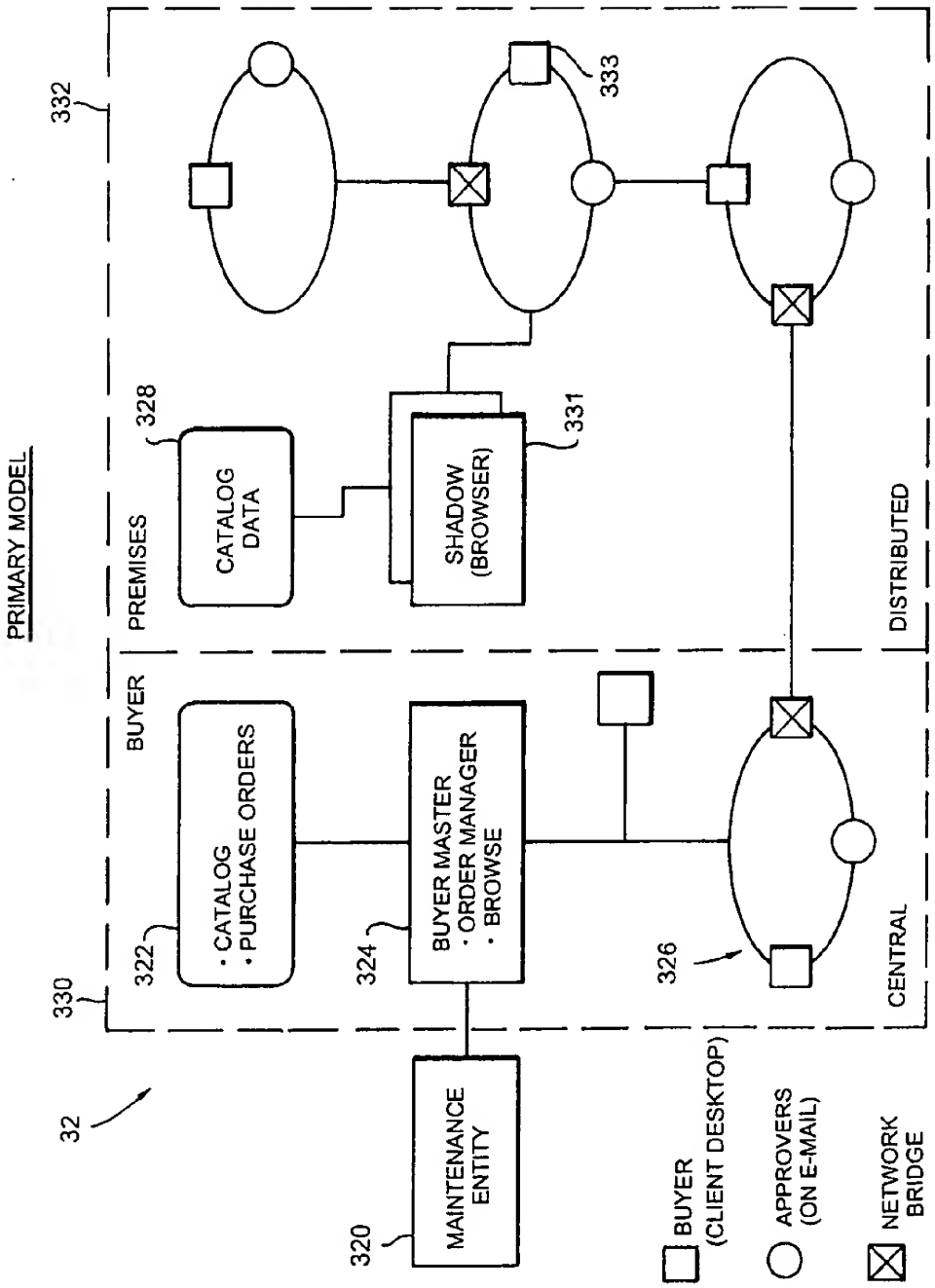

PRIMARY MODEL
PREMISES
DISTRIBUTED
BUYER
CENTRAL
332
333
328
CATALOG DATA
SHADOW (BROWSER)
331
· CATALOG
· PURCHASE ORDERS
322
BUYER MASTER
· ORDER MANAGER
· BROWSE
324
326
330
32
MAINTENANCE ENTITY
320
BUYER (CLIENT DESKTOP)
APPROVERS (ON E-MAIL)
NETWORK BRIDGE

Fig. 12

SYSTEM FOR ORDERING ITEMS OVER COMPUTER NETWORK USING AN ELECTRONIC CATALOG

FIELD OF THE INVENTION

This invention relates to a system for electronically linking a buyer and a seller in a purchasing cycle.

INTRODUCTION TO THE INVENTION

Current procurement procedures in corporations of all sizes are largely manual, or at best semi-automated by electronic mail, and are as a consequence labor-intensive and costly. Typical procedures operate as described below. Refer to the attached drawings, which make the description easier to follow:

1. Typically, an employee requiring some item (such as a piece of office equipment) will look for it in a supplier's catalog. Such catalogs may be at the user's workstation, but are often kept in some central location for general reference (see FIG. **1**, numeral **10**).

2. The employee transcribes information (such as part numbers and price) from the catalog on to a purchase order (FIG. **1**, step **01**).

3. The completed purchase order then normally goes through an approval process, which may, depending on the value of the order and the nature of the items on it, require several sign-offs by people within the corporation with budgetary responsibilities (FIG. 1 step **02**).

4. The purchase order is then sent to the purchasing department of the corporation, which checks the information. These checks include verifying that the items are being ordered from the correct supplier: the same item may be offered by more than one supplier, but contracts are frequently negotiated that require a particular item to be procured from only one of these suppliers, and the Purchasing department has to verify this (FIG. 1, step **03**).

5. The Purchasing department then sends the Purchase Order to the supplier (FIG. **1**, step **04**).

The process described above can take weeks if not months.

At the other end of the procurement process, printing and distributing the traditional paper catalog is also a labor-intensive and costly process (see FIG. **2**, numeral **12**), especially since catalogs are usually of high quality in terms of art-work and color reproduction (FIG. **2**, step **01**) but may be distributed at no charge to potential buyers. The long production time for a typical catalog (of the order of several months) causes additional problems when it includes items such as personal computers and consumer electronics where prices are highly volatile. Vendors have sought a number of solutions to this problem, the most common being to omit prices from the catalog altogether and issue a separate list of prices (FIG. **2**, step **2**) several times throughout the lifetime of the catalog.

In response to this situation, we now disclose a novel system for ordering items. The system comprises:

1) means for receiving and processing images and text from a plurality of catalog content providers for creating and maintaining one or more electronic catalogs in a central location for subsequent distribution over a computer network;

2) means for receiving supplier's price and catalog changes and propagating them to one or more selected buyers over a computer network;

3) a first end-user computer system comprising user interface and able to access disk storage on a shadow catalog server;

4) a shadow catalog server which comprises a second computer system located within the enterprise whose disk storage can be accessed over a local area network by one or more end-users' computers in an efficient manner; said disk storage being being used to hold (1) one or more electronic catalogs, and (2) program code comprising a a "Catalog Browser" capable of transmitting purchase orders to a master buyer and server;

5) a master buyer server comprising a third computer system located within an enterprise containing (1) program code comprising an order manager and a purchase order workflow which takes purchase orders from one ore more end-user computers and and controls their flow through the enterprise's business processes before transmitting them over a network to the supplier; and (2) a purchase order data base.

The advantages of the novel system are appreciated by contrasting it to the prior art summarized above. In particular, the concept of corporations ordering items from suppliers over a computer network is well-established, and has led to the formalization of EDI (electronic data interchange).

The concept of consumers or end-users (the people who will ultimately make use of an item) ordering items from electronic catalogs over a (usually public) network is also well-established. Public network services such as Prodigy (tm), America On Line (tm) and Compuserve (tm) allow subscribers to their services to select consumer and household items from catalogs placed there by suppliers. The items are typically shipped to the subscriber's home, and the cost charged against a credit card. Such systems have to date only allowed the subscriber to access one supplier's catalog at a time.

It may be noted that neither of the above approaches is a complete solution to the problem addressed by the disclosed invention, which is to allow end-users within a corporation to order necessary items as if they were consumers ordering items for their own use and at their own expense, but to have such orders then flow through the enterprise's normal business controls before being submitted to the supplier. The disclosed invention also goes beyond these solutions in allowing the catalog an end-user sees to be sub-setted and otherwise modified from the supplier's general catalog; in facilitating comparisons between items from several suppliers; and in permitting orders to be generated containing items from several suppliers.

BRIEF DESCRIPTION OF THE DRAWING

The invention is illustrated in the accompanying drawings which:

FIGS. **1** and **2** show prior art purchasing and catalog creation processes;

FIG. **3** shows an end-user environment;

FIG. **4** shows a client environment;

FIG. **5** shows a sample screen for features of the present invention;

FIG. **6** shows an invention topological overview;

FIG. **7** provides an overview of data flow between logical servers;

FIG. **8** shows catalog enablement;

FIG. **9** shows content librarian workflow;

FIG. **10** shows image librarian workflow;

FIG. 11 shows a protocol support for filemover; and

FIG. 12 shows a distributed procurement service.

DETAILED DESCRIPTION OF THE INVENTION

The disclosed invention automates much of the manual processing described above. The disclosed invention preferably is used in the following way (FIGS. 3, 4 numerals 14, 16 illustrate preferred steps involved):

1. An employee 17 preferably accesses one or more electronic catalogs 24 stored on a shadow catalog server 22, accessed via a local area network 20 preferably by means of a employee workstation 18. These catalogs contain only those items for which a price has been negotiated between the enterprise and a particular supplier, so the verification by the enterprise's Purchasing department described above is obviated.

2. The employee selects items from the catalogs preferably with a mouse or similar device. Catalog items may be displayed with pictures, descriptions and other information in a fashion similar to a paper catalog. Where similar items are available, a "Compare" icon can be selected on the screen, causing the items to be listed side by side, with differences highlighted. Items can be located by searching down the taxonomy tree of the catalog (much as one searches through a paper catalog by finding the appropriate general section and then looking for a particular item), or by entering a search word or phrase.

3. Items selected may be accumulated in a "clip-board", a temporary holding area on the user's computer disk. When all required items have been selected, the employee selects a "Submit" icon. This causes the selected items in the clip-board to be sent' to the appropriate approvers as a Purchase Order 30. It should be noted that there is no manual transcription of ordering information from the catalog to the purchase order (since that is performed by the disclosed system).

4. After the order has passed through the enterprise's normal (legacy) business systems, including a workflow definition database 26, a purchase order database 28, and other existing corporate applications 32, it is forwarded to the Maintenance Entity via the Network 34. From there it is sent to the supplier for fulfilment in a traditional way.

5. The employee can at any time review the purchase orders already submitted; cancel them if they have not yet been shipped; and print reports.

The other end of this process involves capturing catalog content (text and images as well as price information) in electronic form. As long as catalog content providers and suppliers continue to publish traditional paper catalogs side by side with the electronic versions, this activity does not displace the current manual process.

The two ends of this automated solution, the supplier side and the buyer side, may be brought together by means of a computer network and associated service offerings. FIG. 6, shows an overall network topology 20 or the initial implementation, with the three major pieces (supplier, buyer and Network Central) clearly distinguished. Suppliers 50 provide e-mail directly to the maintenance entity 100 and to the communications gateway 82 via an EDI mail box 110. Suppliers also provide CD's and diskettes for the to terminals 52 and 54 for the purpose of catalog creation and maintenance. Terminals 53 and 54 communicate with the maintenance entity through Token Ring THORNWOOD 56.

Maintenance Entity 100 consists of a local area network backbone 98 which supports multiple token rings 76 and 94 in addition to a communications gateway computer 82. Token ring 76 services catalog file and reference services computer system 72, catalog and software distribution server 74, and PS/2-OS/2 78. Token ring 94 services system and file monitor server 96 and PS/2-OS/2 92. The local area network backbone 98 further supports and EDI gateway pricing decision computer 80. Catalog maintenance activities are input to the Maintenance Entity 100 from remote terminals 52 and 54 via token ring 56 through T-1 leased lines 62. Application development occurs remotely, item 58, and is communicated to the operations environment through application development Token Ring TORONTO 60 and routed to development server 84.

The client environment is shown in the lower segment of FIG. 6, defined by shadow server 106 which maintains a customized copy of the master catalog for distribution to local clients 102 and 104. Purchase orders are received by a Local buyer master server 86 from a data pathway connecting remote shadow LAN 108 with local buyer master LAN 88. The Buyer Master Server also performs the server function in the following capacities; order processing from buyer clients 90, approval and call back. The Buyer Master Server communicates with the operations environment of the enterprise through a 56 Kb switched or leased TCP/IP line 107.

The Buyer Master Server also interacts with the Range of Legacy Systems which consists of a host computer 64 connected to a COMM FEP 66, a MINI computer 68, and a PC 70. The services provided by the Maintenance Entity distinguish this invention from the electronic catalogs offered by public networks such as Prodigy (tm) and Compuserve(tm), which do no more than route messages from the subscriber to the supplier. The Maintenance Entity services act as a single point of contact to all the suppliers on one side and all the buyers on the other. Existing networks merely link them in a many-to-many relationship. The advantages of this approach are clear: for example, a supplier has only to provide a change to a catalog item once to the Maintenance Entity, which then disseminates it to the affected users.

This aspect of the invention is summarized in FIG. 7, numeral 22 which shows the data flows already mentioned in FIGS. 3, 4 as well as some others which were omitted from the original description for the sake of clarity. The omitted data flows include details specific to both the Operations environment and the Client environment:

1. Details of the Operations Environment 125 Networking software and services software including; a PRICING DAEMON 132 which receives purchase orders, change orders and cancellation order acknowledgments from the ORDER PROCESSING SVR 154 located in the Client Environment 123 via an EDI GATE Way 130. The Pricing Daemon 132 in turn provides pricing updates and base catalog entries to catalog file servers, CAT FILE SERVERS 140.

The operations environment further includes distribution management tools, defined generally by CMVC 134. The CMVC consists of a Subscription List 138 which resides on a server in the form of topology tables 136. The subscription list sources multiple reference servers 142, all of which are contained within a distribution server 144. The reference servers source a combination server comprised of a system monitor server 146 and a file mover server 148.

1. Details of the Client Environment 123

Comprised of a Shadow Server 150 consisting of Browser Dynamic link libraries DLLs 152. The Browser DLLs receive catalog data from the Order Processing Server 154

and in turn output the Browser DLLs and customized catalogs, during a client browse session to a buyer (client) **156**.

The Order Processing Server receives inputs from four separate sources; (1) Buyers (clients) **156** (2) the Approval Server **158** (3) the CallBack Server **160** which services the transfer of files to and from legacy systems **164** and (4) the File Mover Server **148**, which is part of the Operations Environment.

This aspect of the invention preferably comprises (see FIG. **7**) three major components:

1. Catalog creation and maintenance tools (shown at the top of Fig. **7**). Catalog creation is defined by item **122**, the SELLER AND PROVIDER ENVIRONMENT consisting of EDI MAIL BOX 122, CONTENT PROVIDER **124**, and CD's & Diskettes **126**.

Catalog maintenance is defined by item **127**, CATALOG MAINTENANCE ENVIRONMENT, which includes item **128**, CATALOG MAINTENANCE CLIENTS which receives inputs from CDS & Diskettes **126** and additions and changes concerning catalog entries & update, pricing updates, and subscriptions from CAT FILE SERVERS **140**.

2. Catalog browsing and purchasing software (the client environment shown in the lower segment of FIG. **7**); and,

3. Networking software and services (the Operations environment shown in the middle segment of FIG. **7**) defined by OPERATIONS ENVIRONMENT **125**.

Catalog Creation Maintenance

The preferred embodiment is made up of two main elements:

Content mianagement tools to receive, process, and manage images **208** and text **212** from content providers **200** for the creation of an EPS (Electronic Purchasing Service) master catalog. An overview of this process is shown in FIG. **8**, numeral and Text **212** from content provides **200** are first converted through conversion units **210, 214** also, including conversion units, **218** and **222** from third party converters **202**, the graphics and text are then and combined with content from independent image providers **220** to create catalogs **216** and **224** constituting third party catalogs **204** which are then combined at an EPS catalog stage **206** to form EPS (Electronic Purchasing Service) catalog **226** and distributed to buyers **230** via EPS subscription **228**;

These enable EPS Operations staff to create and manage catalog information in the merchandise database such as the price, description, and visual representation of each item.

Distribution management tools to receive vendors' price and catalog updates, as well as to propagate the changes to the customers' Buyer Master servers.

Content Management Tools

The EPS Content Management tools preferably comprise:

FotoFarm;

Product Editor;

Folder Editor;

Subscription List Editor;

NAM2DAT/NAM2GRP.

FotoFarm

This collection of utilities may be used to convert text and images from the content providers **200, 250** and **280**. The workflows of these two activities are shown schematically in FIGS. **9, 10** numerals **26, 28**. Supported functions may include:

Receive, store, and archive source images **282** and text files **252** and **282**.

First-level validity check of source media **254, 284** and **286**.

Assign EPS unique filename and update the index files **258, 284**.

Create master catalog's subchapters and folders, and populate them with the relevant contents **260 292**.

Trigger down-stream re-creation or subscription catalogs (see below) when EPS catalog updates occur **260 292**.

Process images received from content providers in batch model **256**:

Delta cropping of image by specifying new crop coordinates **288**.

Generate multiple resolution versions of images.

Convert 24-bit to 8-bit dither with palette matching.

Enable re-scheduling of batch process for related information.

Allow multiple images to be proofed at the same time.

Manage registries of **260 292**:

Shared disk storage for various purposes;

Providers of images, text, or EDI content and services;

All top-level books in the EPS Catalog Server.

Product Editor

This ITS (Iterative Transaction Systems) application enables EPS Operations staff to:

View multiple product descriptions at a time;

Associate images with product handle;

Save, import, and create templates;

View and edit product descriptions.

Folder Editor

A sample screen from this tool is shown as FIG. **11**, numeral **30**. This tool enables Operations Staff to:

View master catalog space.

Organize and arrange products into groupings, i.e., manipulate Catalog topology.

Search: Keyword, Power Scarch (Attribute, Taxonomy).

Subscription List Editor

This uses code from the Folder Editor and Search Engine, with additional functions, to enable Operations staff to:

Save subscription profiles;

Update/edit Distribution Frequency via manual operation;

Update/edit Distribution Frequency automatically;

Generate flat or hierarchical browser space;

Send newly created subscription list to Catalog Server.

NAM2DTA/NAM2GRP

NAM2DTA is a stand-alone GML parser that converts tagged source files to catalog objects. NAM2GRP is a stand-alone GML parser that converts tagged source files to catalog folders.

Distribution Management Tools

Distribution Management concerns itself with getting data from Network Central out to the customers' computers on the network in such a way that each computer receives only that data (catalog items and price information) which it needs. The EPS Distribution Management tools preferably comprise:

Distribution Manager with GUI query tool;

Catalog daemon (CATD);

FileMover.

Distribution Manager
Responsible for
Price Update (Catalog Monitor);
Catalog Update (Catalog Monitor);
Automated EDI Processing and Distribution (Update Daemon);
Acknowledgement Processing;
Update DB2/2 Tables.
Catalog Daemon (CATD)
This software runs in customers' servers and polls mailboxes to apply updates, and preferably monitors channels for action objects including: Images; Applications; Prices; Catalog descriptions. It preferably can Execute action specified in action object; Forward acknowledgement objects to parent; and is Used together with FileMover daemon to verify file movement.
Filemover 300
This is the communication layer of EPS which moves files between systems. In particular, it drives the movement of data throughout the network for price up-dates and purchase orders.
Filemover uses a simple mail-like mechanism in which files are moved from OUTBOX directories to INBOX directories, which can span network attached systems. It has no knowledge of the type or structure of the file it moves, and treats them all as binary blob data. Hence no code page translation or character set translation is attempted.
Filemover is a Point-to-Point protocol and does not provide any internal routing mechanism. Routing can be provided outside of the Filemover, however, by:
Utilizing the routing mechanisms of the underlying protocols, e.g., TCP/IP. Will only work across machines running in a single internetwork with the same protocol.
Using Catalog Daemon (CATD) to provide routing by forwarding files across intermediate nodes. It works across internetworks with different communication protocols but its network path must be hard-coded within the catalog files being sent.
Filemover enables the EPS to:
Move files of any size and takes care of splitting and reassemblying the files when the underlying communication software has a limitation on the file size.
Verify and confirm both file movement and ability to move files (e.g., checks disk storage).
Support file movement over systems connected with multiple protocols (shown schematically in Fig. 11):
IPX/SPX **306**
NetBIOS **304**
TCP/IP **310**
Support file movement over:
Dial-up connection (SLIP and LAN Distance **312**) via modems
Dedicated LAN or WAN connection
Support Store and Forwarding of files **302**.
SNA APPC **308**
Client Environment
Recall that the Client Environment (FIG. 7) comprises two principal components:
1. An electronic catalog in a format that can be browsed, searched and ordered from, by a corporate employee with no training in Purchasing procedures;
2. Software that controls the flow of a purchase order through an enterprise's procurement procedures.
The preferred embodiment of the above software and related manual processing minimizes data storage requirements and maximizes the responsiveness of the total system preferably by employing a primary model consisting of two major components, a buyer side component **330** which communicates over a network bridge with a premises component **332** (see FIG. **12**, numeral **32**):
1. An end-user computer system **333** attached to a Local Area Network or LAN, containing the usci interface;
2. A "shadow catalog server", **331** (FIG. **12-2**) with disk storage that can be accessed over a LAN by one or more end-users' computers, the disk storage being used to hold one or more electronic catalogs **328** and program code **331** to enable browsing of the catalog and transmitting purchase orders to the "buyer master server" **324**;
3. A "master buyer server" **324** (FIG. **12-3**), which is a computer system within the enterprise containing (1) program code (described below) which can take purchase orders **332** from one or more end-user computers and control their flow through the enterprise's business processes, as described under "Workflow" below, before transmitting them over a network to the supplier via the Maintenance Entity **320** and (2) a to a Purchase Order data base **322** that can be accessed over a LAN **326**.
A preferred embodiment of the above comprises:
Order Manager and Catalog Browser
This function runs on the end-user's personal computer, although the code would normally reside on disk storage in a catalog shadow server machine. It provides the following main function to an employee using the system:
Log on/Password Security
Login
Log into EPS
Track User ID for all transactions arising from this session.
Additional Order Manager functions may be enabled or disabled based on the login profile.
User selects from list of authorized users.
Catalog Browser
Browse Product Images, Text and Prices
Able to page forward or backward.
Quick return to top menu page from any part of the catalog.
Quick return to the table of contents from any part of the catalog.
Display previous page at top of screen, with links to navigation log.
Images are displayed in .BMP format.
Two separate image files are kept for OS/2 and Windows. See also "FotoFarm", supra.
Text The Browser may select zero, one, or more ordered sets of descriptive phrases.
Prices.
Select Product Based on Single Keyword
Based on index search.
Index search is launched with user's action on an icon represented by a magnifying glass.
Search by product type or manufacturer's name.
Copy to clipboard for further processing.
Compare Products (max. 4)
Compare up to four items from the same category.
End user selects this option by clicking on a "Compare" icon represented by a scale.
Varying features amongst the compared products are highlighted in bold text.

Product Clip Board

Select items on Product Listing for adding to clipboard.

Add item on Product Page to clipboard.

Delete an item in the clipboard.

Change the quantity of an item in the clipboard.

Clear the clipboard to remove ALL items.

Save the clipboard (to a file).

Submit the clipboard (as a purchase request).

Show the items on the clipboard.

View clipboards (i.e. saved clipboard files).

Purchase Request Generation Select the recipient of the purchased items from a list. The recipient list is kept in a local disk, and its entries can be added, changed, or removed.

Classify all line items into capital and expense items.

Ask for budget number for capital items.

Ask for engagement number for expense items.

Display generated purchase request for confirmation.

Select approver from list to submit request to.

Add comments to purchase request.

Send purchase request to approver.

Save purchase request as clipboard.

Cancel submission of purchase request.

Print Clipboard: This function is in addition to, and separate from, the report generation functions which use DB2/2 report generators. It enables users without access to DB2/2 to print a clipboard or submitted order from their own workstations.

The supported functions include:

Generate printed output from client application for the contents of the clipboard.

Can also be used for printing a submitted order.

Able to generate output for various printer formats including Postscript. May have to go through some type of metafile generation process.

Needs to work in both OS/2 and Windows.

Purchase Order Creation

Multiple Sellers on One PO

Each line item in a purchase request could be sent to a different vendor. This requires that information such as the shipping and billing address be stored on a line item level, rather than at the header level for a purchase order. These multiple sellers include those whose products are not listed on the catalogs.

Electronic PO

This is to forward the purchase orders electronically to the vendors via the EPS. system. Data includes type of transaction, required data as defined by EDI standards for a **850** PO such as PO number, date, name & address, customer ID, customer master record for shipping and billing information.

Print Paper PO

Print PO by vendor in a multi-vendor PO

Sub-function of the common printing functions described in "Report Generation", infra.

PO Processing

Manual Status Update

Purchaser can update status of PO manually after receiving acknowledgements, status updates, etc. from vendors via fax, phone, or mail. Changes to the PO can then be saved to the DB2/2 database on the Purchasing Server.

Line Item Modifications by Client

The quantity of line items can be increased or decreased prior to order submission. After an order has been submitted, the quantity can only be decreased.

The quantity of line items can be decreased to zero.

A line item can be deleted once an order has been placed with a vendor and is being fulfilled by the vendor.

Allow client to modify other fields such as requested ship date, shipping and billing address, add comments to line items (e.g., a banking institution).

Possibly allow client to switch partnumbets, delete line items, add new line items.

Change Logging/Reporting

Changes to the POs are recorded in the logs and can be accessed by the report generation functions.

PO Maintenance

Browse POs

Group existing POs in ciapteus with summary information including:

Request number.

Requester.

Recipient.

Request date.

Total price.

Line of Business.

Scroll through all line items.

Sort line items by column headings in the following order:

Sort numeric columns from high to low or low lo high.

Sort alphabetic columns from A to Z or Z to A.

View details of line items by clicking on them.

Search for specific groups of POs and purchase requests by Requester Name, Requester Date, and Request Number. The search results can be grouped into a chapter.

View Approvers

Enables user to look up list table of associated approvers for a PO.

Approvers are item-based; i.e., each item has its own approver.

Enables user to view approval data for each item, with the approvers name, decision, and date of action.

Check Status

Enables users to check current status of POs. * When orders are placed, vendors send acknowledgements and status messages via EDI. These are reflected in the updates to the status of line items, with the date of the status change.

The approval status of each order or request can also be checked.

Cancel POs

Enables users or administrators to cancel POs or purchase requests. This is dependent on whether the order has passed the deadline for the change to be effective. Vendors restrict the set of order states against which a PO can be canceled. For example, if the item has been shipped, the order cannot be canceled. EDI Vendors must be able to support Cancel/Change **860** transactions and their subsequent acknowledgements.

The end user will be prompted for confirmation of cancellation request prior to processing.

Upon confirmation of cancellation, the order is moved to the Canceled Requests chapter.

Note

If the PO has already been sent to the vendor, an additional EDI (**860**) transaction is generated and budgets credited.

Add Approvers

This function is required for customers who are not using e-mail approval.

Requires end-user interface to be modified to make the approver line items editable.

Changes must be saved back to the database.

Delete Approvers

This function is required for customers who are not using e-mail approval.

Order Manager Admistration

Budgets

All line items for capital purchases are charged against the capital expense budgets.

Budget Information: Each budget has the following information:

Budget number

LOB (line of business)

Line of business related to the purchase.

Allocated

Amount allocated to the budget.

Current Requisition

Amount associated with all active orders.

Ordered

Amount for orders that have been sent to vendors.

Balance

Balance amount available in the budget.

Active

Yes for an active budget.

No indicates a budget that has been closed and no expenditure can be charged against it.

Create new capital budget: The following information is preferably required:

Budget number

Amount allocated

Amount allocated to the budget.

Active status

A Yes to indicate that it is active.

LOB (line of business)

Line of business the budget is assigned to.

Description of budget

Brief description of the purpose of the budget.

Change existing budget: Changes can be made to the following aspects of the capital expense budget:

Amount allocated

Amount allocated to the budget.

Active status

A Yes to indicate that it is active.

Description of Budget

Brief description of the Purpose of the Budget.

All changes are logged to the change history log, which can be viewed.

Delete existing budget: This function may be used to clear existing budgets at the end of a fiscal year.

There will be a prompt for the user to confirm deletion of the budget amount.

View budget history

All changes to the allocated amount and status of the budget, as well as all expenditures are logged.

Each transaction is logged with a brief description of the activity attached.

The last 100 transactions are logged.

This view function is applicable to both active and inactive budgets.

Import budget guidelines: This is to enable the import of budgets from SAP, or another accounting interface, for the initial budget creation.

Track budget: All line items in a PO get charged to either an expense or capital budget. Any modifications to the PO requires that the budget balances get updated.

User Profiles

The following functions are provided:

Add new users Authorize new users to the list of authorized users.

Delete User

Update current user profile

Ship/Bill Update (Per Site)

This is to enable the purchasing administrator to maintain the shipping and billing addresses of all locations within a Customer enterprise. These addresses are referenced by POs to define the shipping and billing destinations. The supported functions include:

Add new address;

Update current address;

Delete current address;

Vendor Info Update

This is to enable the purchasing administrator to maintain the EDI vendor addresses for confirmation of shipment and billing, as well as status updates. The supported functions include:

Add new address;

Update current address;

Delete vendor address.

Define Company Policy For Captial/Expenses Purchases

Line of Business

Add new line of business;

Update line of business information;

Delete line of business.

Report Generation

This includes the common printing functions for both the reports and purchase orders. They include:

Commodity reports;

Line cost reports;

Order modification reports;

Budget modification reports;

Customized reports.

Approval Workflow

Approval workflow is controlled by the Approval Manager residing in the Purchase Server in the customer's site. This workflow of the purchase orders between the customer and vendors is enforced by a PO approval process defined by the customer. Its functions include:

Keep track of a PO's approval status from the moment a purchase requisition is generated. Appropriate actions are taken to forward the purchase requisition to the predefined approvers to be approved or rejected.

Interface with the customers' electronic mail systems to post approval notifications for the necessary action by designated PO approvers.

Provide separate ITS client application to allow PO approvers to approve purchase requisitions directly from within the EPS system rather than from the external email system.

Approval Policy Configuration

Set up Lotus Notes DB to specify approver hierarchy.

Use of REXX code to customise approval hierarchy.

Approval Data

Store approval data for POs in DB2/2 PODB;

Store list of approvers in Lotus Notes;

Entry point API (call-out) to support accessing approvers from external systems.

Approval List Generation

Print approver Lists from Lotus Notes;

View approver Lists from Lotus Notes;

Lotus Notes interface to create and update approvers list (company wide);

REXX code to specify approval policy given PO and list of approvers;

Generate Approvers service application **158** to communicate with client server API CORDER PROCESSING SVR **154** (per PO).

Approval List Processing

Approvers can receive email messages via an E-MAIL SERVER **162** notifying them that they need to approve various purchase requisitions.

Mail routing support is available for Lotus Notes, cc:Mail, and Microsoft Mail.

Snapshot Database Within Lotus Notes

This is a function to synchronise the information between the EPS Server and the Lotus Notes server.

Help Functions Within Lotus Notes

A help database is available from within Lotus Notes as a Notes database.

Approval Manager Client

This ITS client application is for approvers who do not want to receive approval notification from an email system external to EPS. It enables approvers to log on to the EPS Purchasing Server and approve or reject purchase requisitions.

PO Workflow

The flow of the purchase order through an enterprise's approval and other financial processes varies with each enterprise. The disclosed system contains workflow logic implemented as a Finite State Machine. This is a table specifying how the system is to change state in response to specified inputs, and what actions it should take when each transition takes place. Such a table can be easily tailored to fit the needs of a particular enterprise. Application Program Interfaces (APIs) in the generic state transitions supplied with the system allow an enterprise to invoke and pass information to and from existing computer applications and data bases (which could include the enterprise's "legacy" purchasing system) as shown in FIG. **4** step **02**.

In the preferred embodiment, The EPS Client/Server application programming interface (API) provides client applications with a set of functions and action calls to communicate with the EPS Server for managing purchase orders within the customer's environment. It can also be used by any customer applications to work with the data available from the server.

The API supports three types of client applications:

Interactive Transaction System (ITS) applications

These are clients written with the ITS toolkit and using the ITS runtime to provide the user interface.

Non-ITS applications

These are clients that have user interfaces other than ITS.

EPS system extensions

These are ITS and non-ITS applications registered with the Purchase Order workflow of the EPS server, and can be classified into two categories:

1. EPS Monitors

Registered against a certain state in the Purchase Order workflow and are notified whenever any purchase request enters that state. The notification will be received asynchronously with the purchase request continuing within the workflow.

2. EPS Services

Registered with the EPS Server and introduce a new state in the Purchase Order workflow. They are notified whenever any purchase request enters that state, and are expected to notify the EPS Server when the specific task is completed.

API Architecture

A brief description of the API architecture is necessary for an understanding of the API functions and action calls. Essentially, the API is made up of five layers, as shown in FIG. **5**:

X layer **38** This is the ITS application layer which interfaces with ITS client applications to deliver ITS client application requests to the C layer for onward transmission to the Buyer Master.

Y layer **40** This is the non-ITS application layer which interfaces with non-ITS applications to deliver non-ITS client application requests to the C layer for onward transmission to the Buyer Master.

C layer **42** This is the common layer that interface the X and Y layers with the M layer. It takes application level data structures from X or Y layers, converts them into low level communication data buffers, and invokes M layer functions with these data buffers to communicate with the EPS Server.

M layer **44** This message layer is the lowest level EPS API communication layer. It uses the ITSCOMM **48** API to handle protocol specific communication functions. It handle communications data buffers instead of application level data structures.

S layer This is a server interface layer and handles functions between the C layer of EPS Server and the OM Server layer **47**.

In addition, the EPS Client/Server API has a set of OMServer APIs to support client requested actions on the EPS Server from the S layer **46**.

APIs

Logically, the EPS Client/Server APIs can be grouped into four areas:

Information

These include the functions and actions to Get and Save data of predefined datatypes to the EPS Server.

Monitors

Services

Security

These validate clients and restrict the level of access. These include: Logon, Logoff, Connect, and Disconnect.

THE NETWORK ENVIRONMENT

"Network Central" (shown in the middle of both FIGS. **6**, **7**) consists of one or more computers and program code of the type located in IBM's Advantis network, which provide four principal functions:

1. receipt of purchase orders from the master buyer servers of one or more enterprises;

2. passing EDI transactions to and from suppliers;

3. storing catalog information for transmission to buyers;

4. the distribution of price information.

In addition, personnel at Network Central may perform many or all of the activities described under "Catalog Creation/Maintenance" above.

Overview of EDI

The Electronic Data Interchange (EDI) is a standard for the exchange of business data. It defines:

Communication wrappers, which are usually handled by a communication package from a Value Added Network (VAN) providing EDI mailboxes).

The ASCII character set.

Various transactions, each with an ID

The order and hierarchy of data within each of the transactions.

The type and length restrictions for each piece of data.

There are two major EDI standards—ASC X12 which is the standard for the United States, and UN/EDIFACT, which is the general standard adopted by countries outside of the U.S.

EPS EDI Implementation

Customers may be connected to a system like the IBM &EPS. via non-EDI links, sending their purchase orders over 56khs lines. The EDI Gateway translates and maps these communication into EDT messages before using the FileMover mechanism to send them to vendors via its EDI mailbox on the Advantis network. Vendors' acknowledgements and status updates are transmitted as EDI messages to the Advantis mailbox. The EDI Gateway translates them to the EPS format and updates the PO status accordingly.

Currently, the EDI Gateway consists of the EDILISTS application as well as the IEBASE program provided by Advantis to connect to the Advantis network using LU6.2 communication protocol. The gateway logs and tracks all EDI transactions for diagnostic purposes; it can also alert the Network operator for the necessary recovery action when it encounters any unexpected event.

The preferred embodiment supports the following EDI transactions:

832-Create Catalog Content.

850-Purchase Order.

855-Order Acceptance.

856-Shipping Order Status.

860-Change/Cancel Order.

865 and 870-Order Status.

997-Acknowledge Receipt of Order.

Other Gateway Functions

Gateway EDI in-box and PO in-box monitoring

Look for new PO request and translate to EDI.

Look for new EDI transaction and translate to EPS internal format.

Poll EDI mailbox hourly if not done via PO transactions.

Execute utility to access Advantis's mailbox using IEBASE

Upload EPS-generated EDI to VAN.

Download any waiting EDI message.

Directory browsing.

PO lookup, browsing and merge update. Look up and browse archived PO files.

PO/EDI error logging.

Reports all errors and alerts system operator for attention in the event of a major error taking place.

Logs problems to a flat file.

Event tracking and logging.

Catalog tools linking for 832 transactions.

Archiving POs to multiple customer/vendor file systems.

What is claimed:

1. A system for electronically ordering items within an enterprise comprising:

a maintenance entity. located outside said enterprise, for receiving price and availability data from a plurality of distributors and means for receiving and processing images and text of catalog data from a plurality of catalog content providers for creating and maintaining one or more electronic master catalogs in a central location for subsequent distribution to a plurality of shadow catalog servers distributed throughout said enterprise, where said shadow servers contain data representing a customized electronic catalogue from said master catalogs, over a computer network, said maintenance entity comprises:

means for constructing said customized catalog from said master catalogs;

means for receiving a supplier's price and catalog availability changes from a plurality of said distributors and propagating them to one or more selected buyers over said computer network;

means for receiving catalog changes from a plurality of said catalog content providers and propagating them to one or more selected buyers over said computer network;

one or more computer systems, maintained by said maintenance entity, and having means for creating and transmitting to a plurality of shadow catalog servers within said enterprise, images and text of a plurality of catalog items offered by said content provider and price and availability data from said suppliers; said enterprise comprises:

a plurality of first end-user computer systems geographically distributed throughout said enterprise comprising a user interface and able to access disk storage on said shadow catalog server, and having means for electronically ordering said catalog items from said shadow catalog servers;

said shadow catalog servers, which comprise a second computer system whose disk storage can be accessed over a local area network by one or more first end-user's computers; said disk storage being used to hold and maintain (1) one or more customized electronic catalogs, and (2) internal program code comprising a Catalog Browser capable of transmitting purchase orders to a master buyer and server, and having means for allowing an end user to view said customized electronic catalog in order to facilitate comparisons between catalog items from several suppliers, and in permitting orders to be generated containing items from several suppliers;

said master buyer server comprising a third computer system located within said enterprise containing (1) purchase order workflow software comprising an order manager and a purchase order workflow which takes purchase orders from one or more end user computers and controls their flow through said enterprise's business processes before transmitting them over a network to the supplier; and (2) a Purchase Order data base; and

means for said master buyer server to receive information from and transmit information to said supplier.

2. A system according to claim 1, further comprising a means for specifying, subscription information comprising a mapping between a plurality of supplier's catalog items and one or more buyers for indicating which catalog items can be propagated to each buyer's catalog server, where said mapping is unique to each said buyer's catalog server.

3. A system according to claim 1, further comprising a means for specifying, pricing information comprising a mapping between a plurality of supplier's catalog items and one or more buyers for indicating which catalog items can be propagated to each buyer's catalog server, where said mapping is unique to each said buyer's catalog server.

4. A system according to claim 1, further comprising an EDI gateway wherein EDI transactions between one or more distributors and said maintenance entity are converted to

pricing, catalog availability and purchase order transactions between one or more customers and said maintenance entity, and wherein EDI transactions between one or more suppliers and said maintenance entity are converted to catalog update transactions between one or more customers and said maintenance entity.

5. A system according to claim 1, further comprising a means for enabling said purchase order workflow software to be mapped to the business processes of an enterprise.

6. A system according to claim 1, further comprising mapping tables constructed by said maintenance entity for identifying suppliers, manufacturers, content providers, catalogs, catalog items and buyers for uniquely identifying each, whilst allowing said suppliers, manufacturers, content providers and buyers to continue to use the naming conventions established within their organizations.

* * * * *

US006192380B1

(12) **United States Patent**
Light et al.

(10) **Patent No.: US 6,192,380 B1**
(45) **Date of Patent: Feb. 20, 2001**

(54) **AUTOMATIC WEB BASED FORM FILL-IN**

(75) Inventors: **John Light**, Hillsboro; **John Garney**, Aloha, both of OR (US)

(73) Assignee: **Intel Corporation**, Santa Clara, CA (US)

(*) Notice: Under 35 U.S.C. 154(b), the term of this patent shall be extended for 0 days.

(21) Appl. No.: **09/052,902**

(22) Filed: **Mar. 31, 1998**

(51) **Int. Cl.**[7] **G06F 7/06**
(52) **U.S. Cl.** **707/505**; 707/506; 707/507; 707/508; 707/9; 707/10
(58) **Field of Search** 707/505, 506, 707/507, 508, 9, 10

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,640,577 * 6/1997 Scharmer 395/768
5,794,259 * 11/1998 Kikinis 707/507
5,802,518 * 9/1998 Karaev et al. 707/9
5,931,907 * 8/1999 Davies et al. 709/218
5,963,952 * 5/1999 Smith 707/102
5,974,430 * 10/1999 Mutschler 707/505
6,029,245 * 2/2000 Scanlan 713/200

OTHER PUBLICATIONS

Laura Lemay's Teach Yourself Web Publishing with HTML 3.2. pp. 555,560,561,562,757. Copyright 1996 by Sams.net Publishing, 1996.*

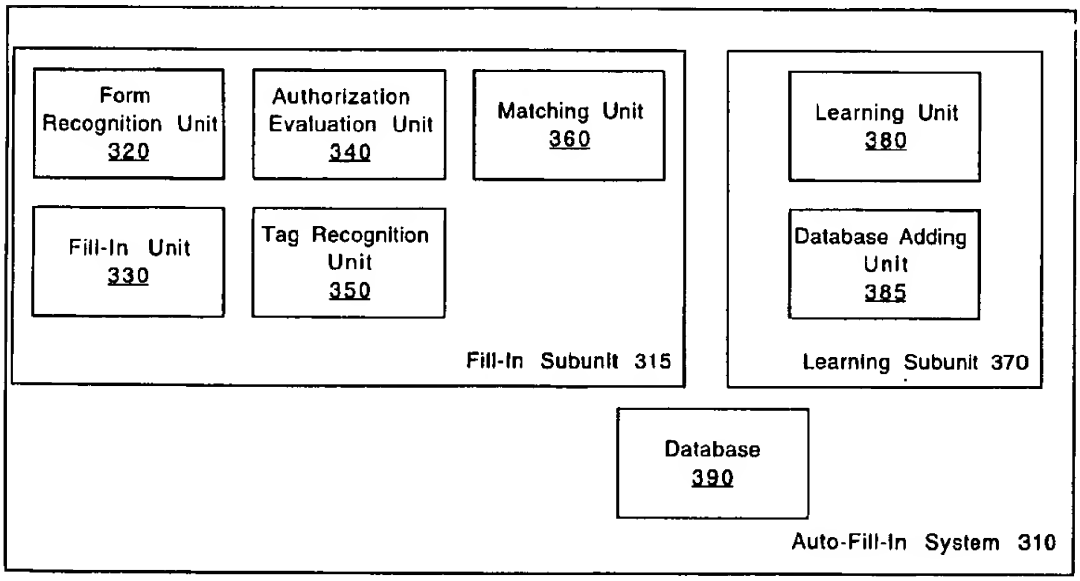
* cited by examiner

Primary Examiner—Thomas G. Black
Assistant Examiner—Thuy Do
(74) *Attorney, Agent, or Firm*—Blakely, Sokoloff, Taylor & Zafman LLP

(57) **ABSTRACT**

A method and apparatus for automatic web form fill-in is provided. A web page is accessed. A form included in the web page is recognized. Data is automatically filled into the form from a database.

19 Claims, 7 Drawing Sheets



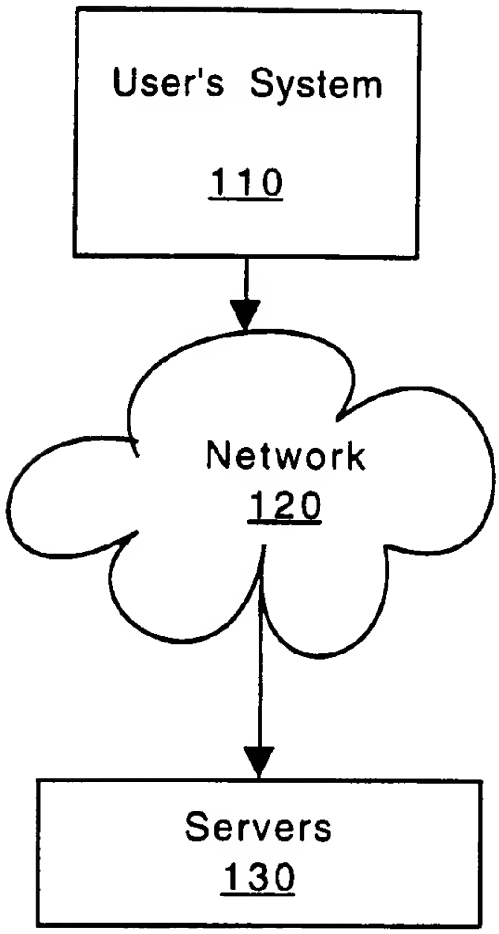


Fig. 1

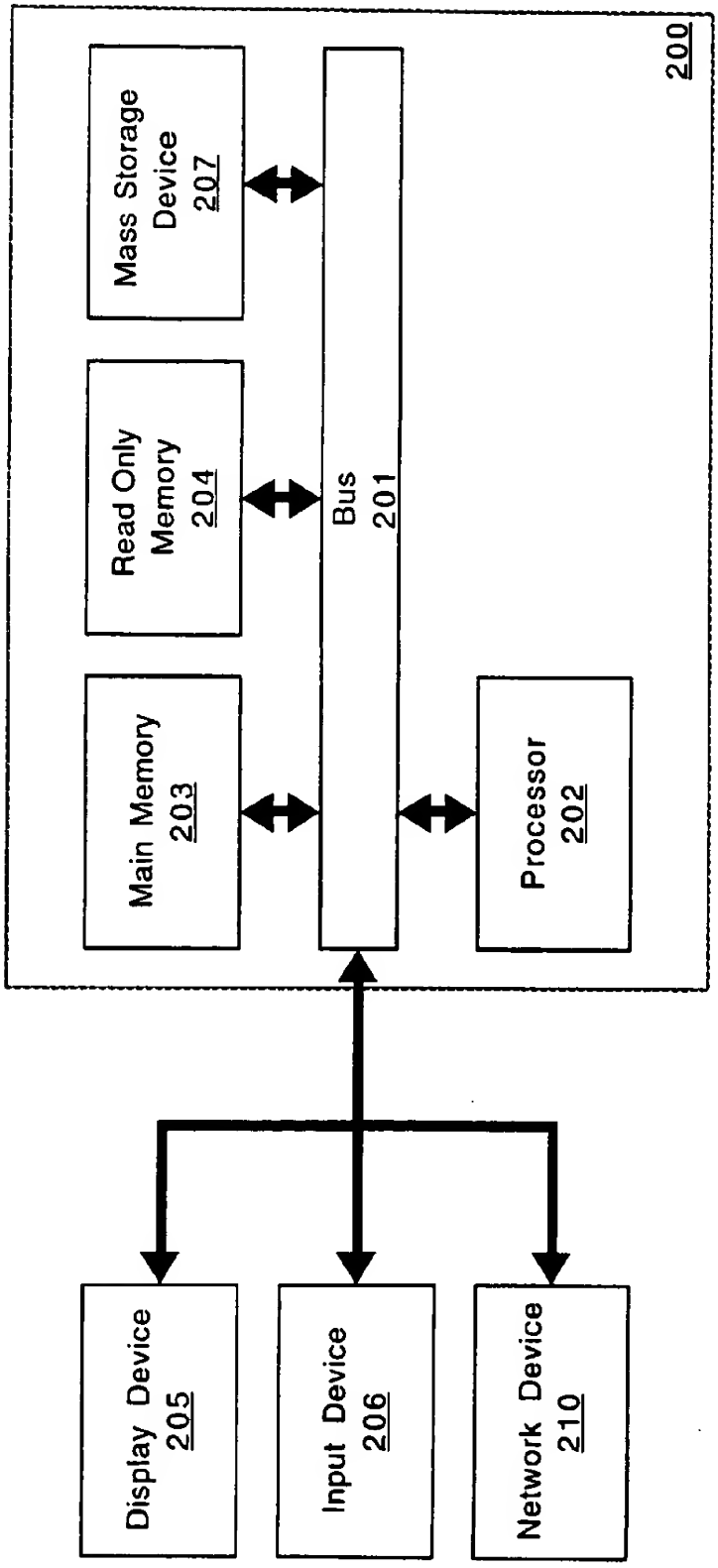

200
Mass Storage Device 207
Read Only Memory 204
Main Memory 203
Bus 201
Processor 202
Display Device 205
Input Device 206
Network Device 210

Fig. 2

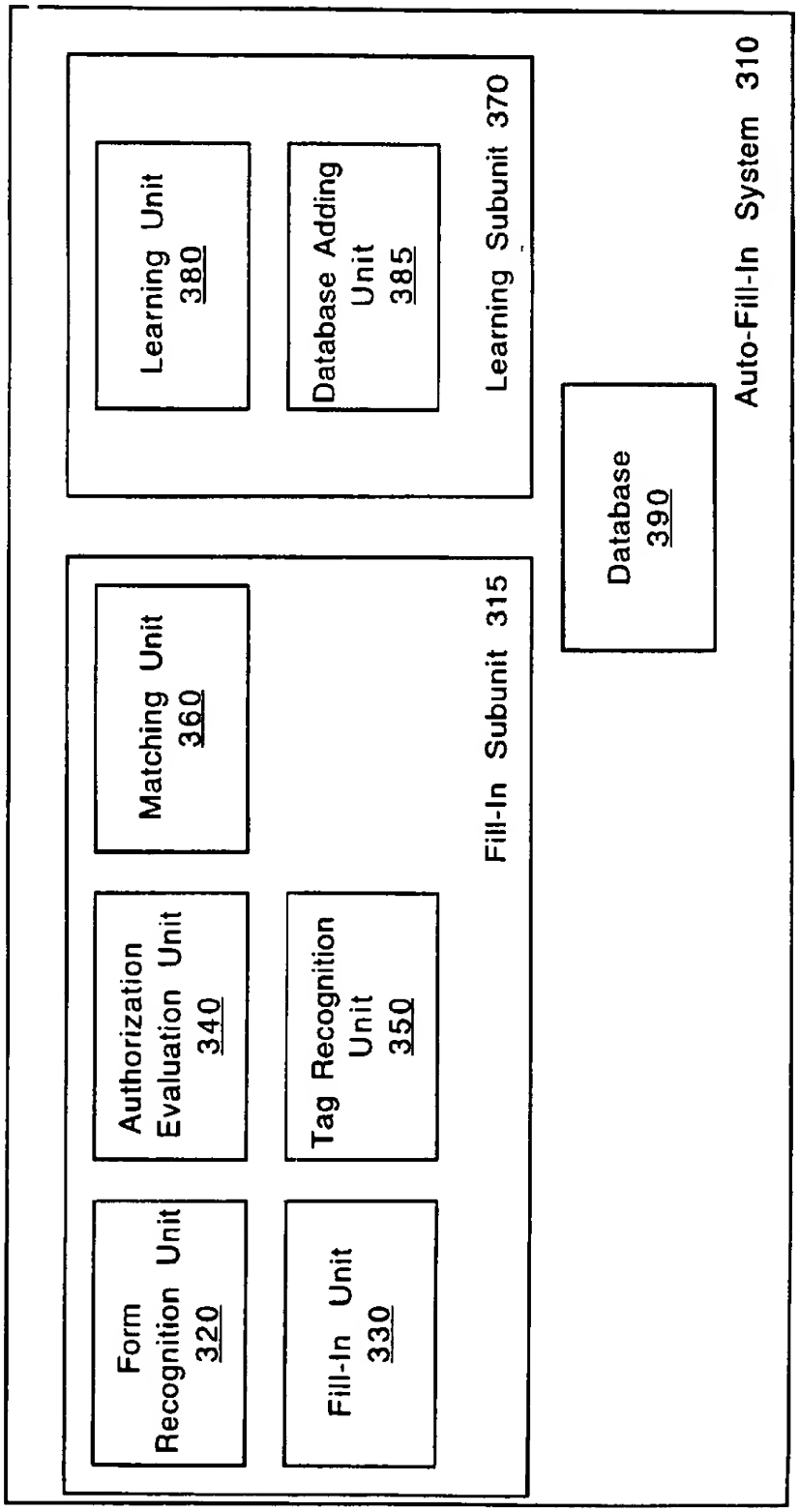

Auto-Fill-In System 310
Learning Subunit 370
Learning Unit 380
Database Adding Unit 385
Database 390
Fill-In Subunit 315
Matching Unit 360
Authorization Evaluation Unit 340
Tag Recognition Unit 350
Form Recognition Unit 320
Fill-In Unit 330

Fig. 3

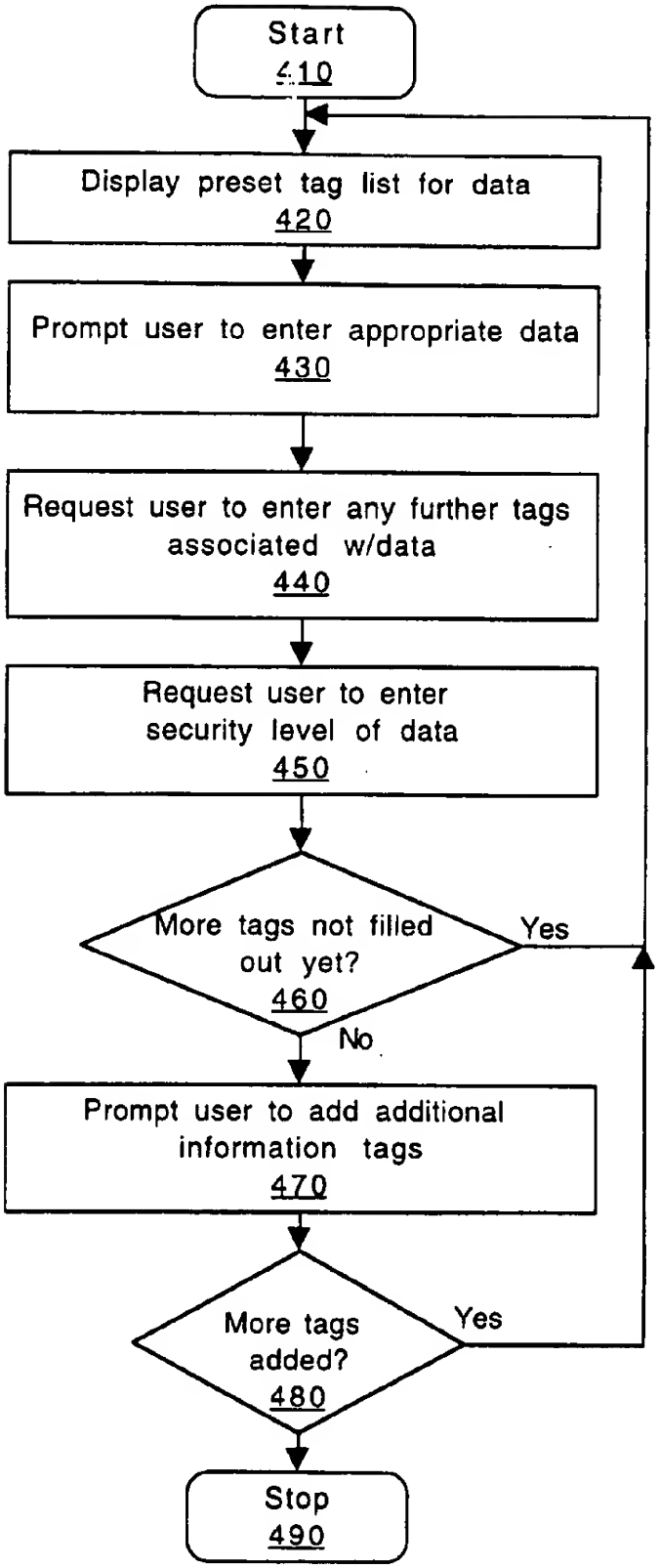

Start
410
Display preset tag list for data
420
Prompt user to enter appropriate data
430
Request user to enter any further tags associated w/data
440
Request user to enter security level of data
450
More tags not filled out yet?
460
Yes
No
Prompt user to add additional information tags
470
More tags added?
480
Yes
Stop
490

Fig. 4

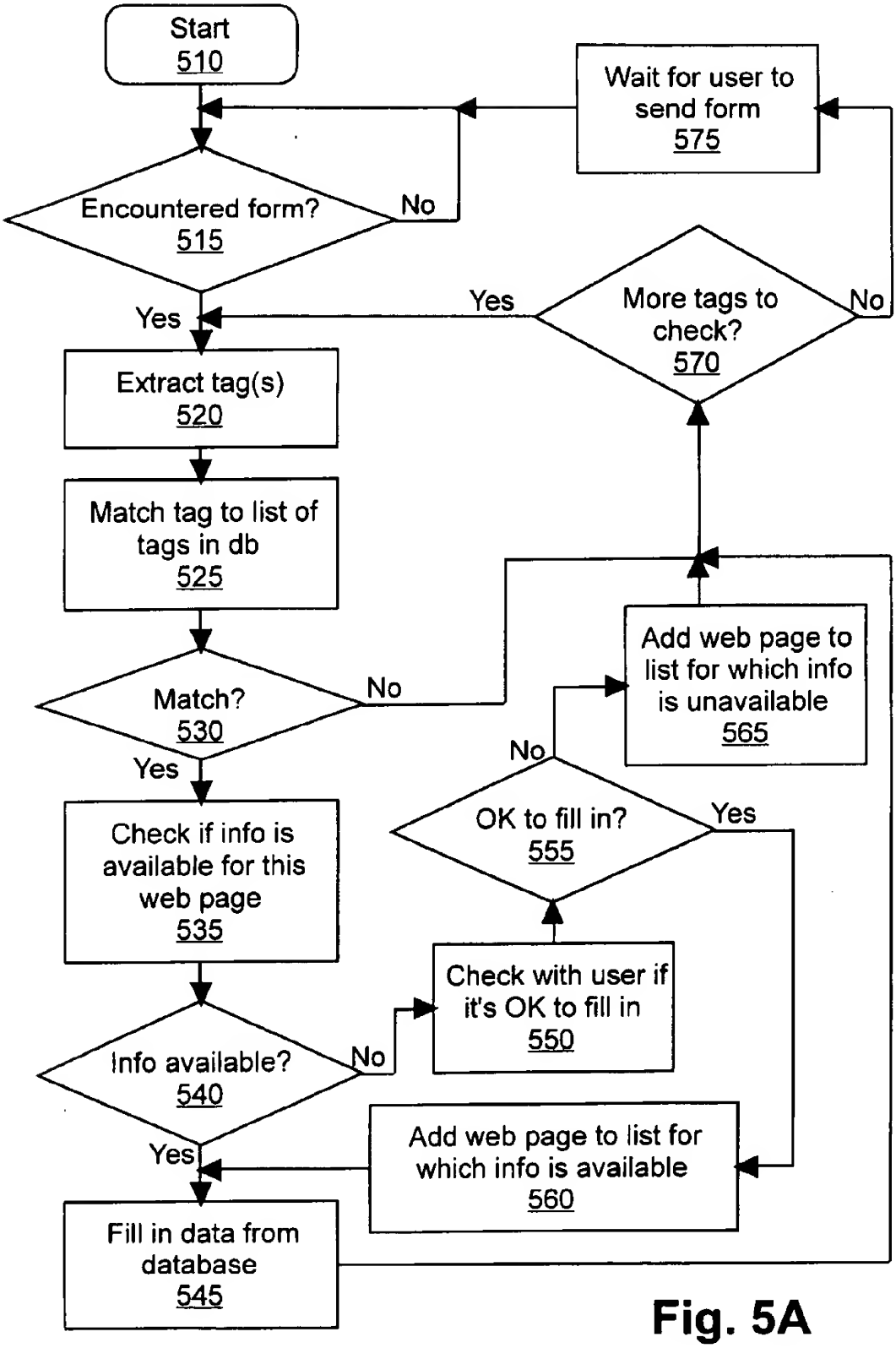

Start
510
Wait for user to send form
575
Encountered form?
515
No
Yes
More tags to check?
570
Yes
No
Extract tag(s)
520
Match tag to list of tags in db
525
Match?
530
No
Add web page to list for which info is unavailable
565
Yes
No
OK to fill in?
555
Yes
Check if info is available for this web page
535
Check with user if it's OK to fill in
550
Info available?
540
No
Yes
Add web page to list for which info is available
560
Fill in data from database
545

Fig. 5A

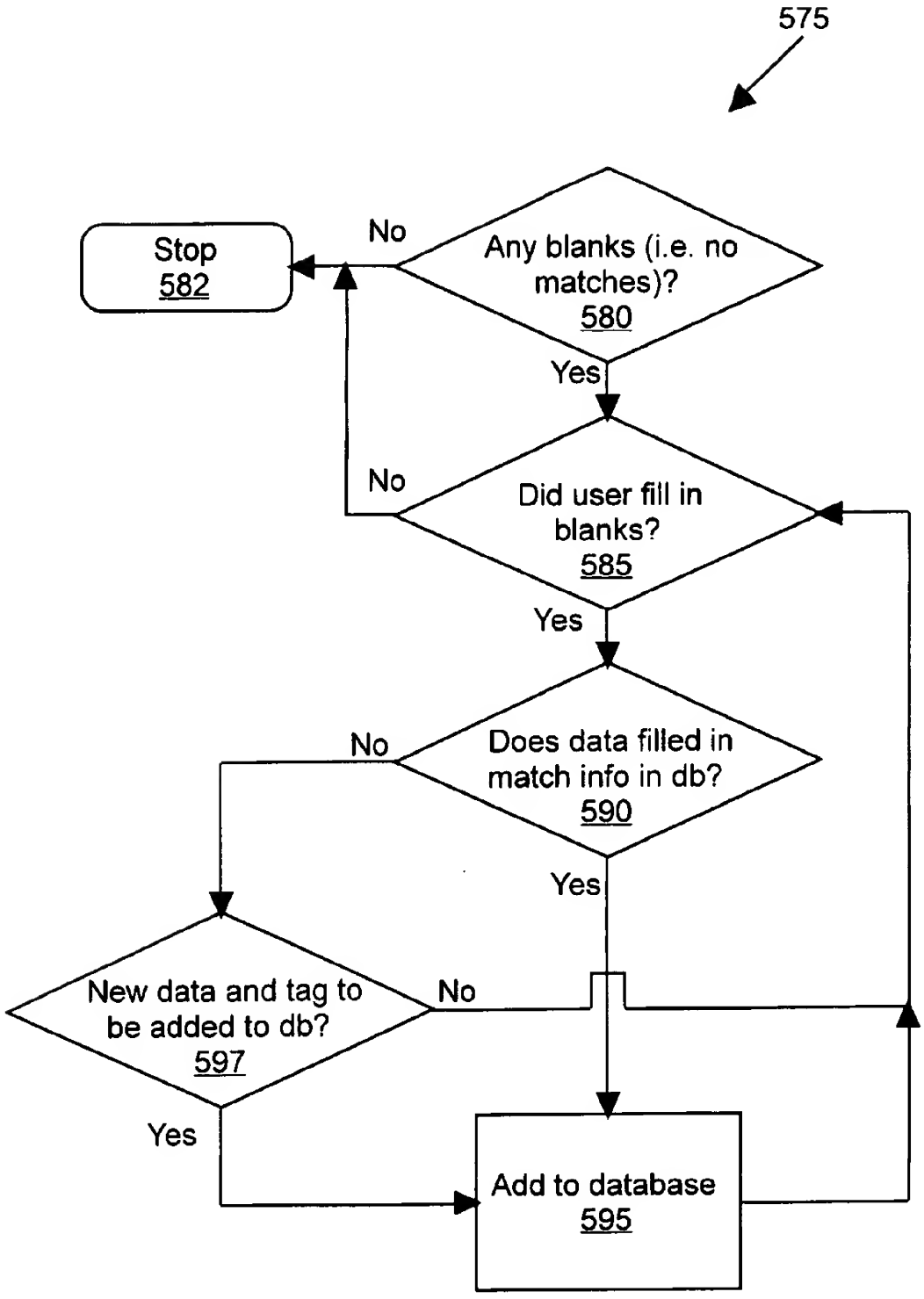

575
Stop
582
No
Any blanks (i.e. no matches)?
580
Yes
No
Did user fill in blanks?
585
Yes
No
Does data filled in match info in db?
590
Yes
New data and tag to be added to db?
597
No
Yes
Add to database
595

Fig. 5B

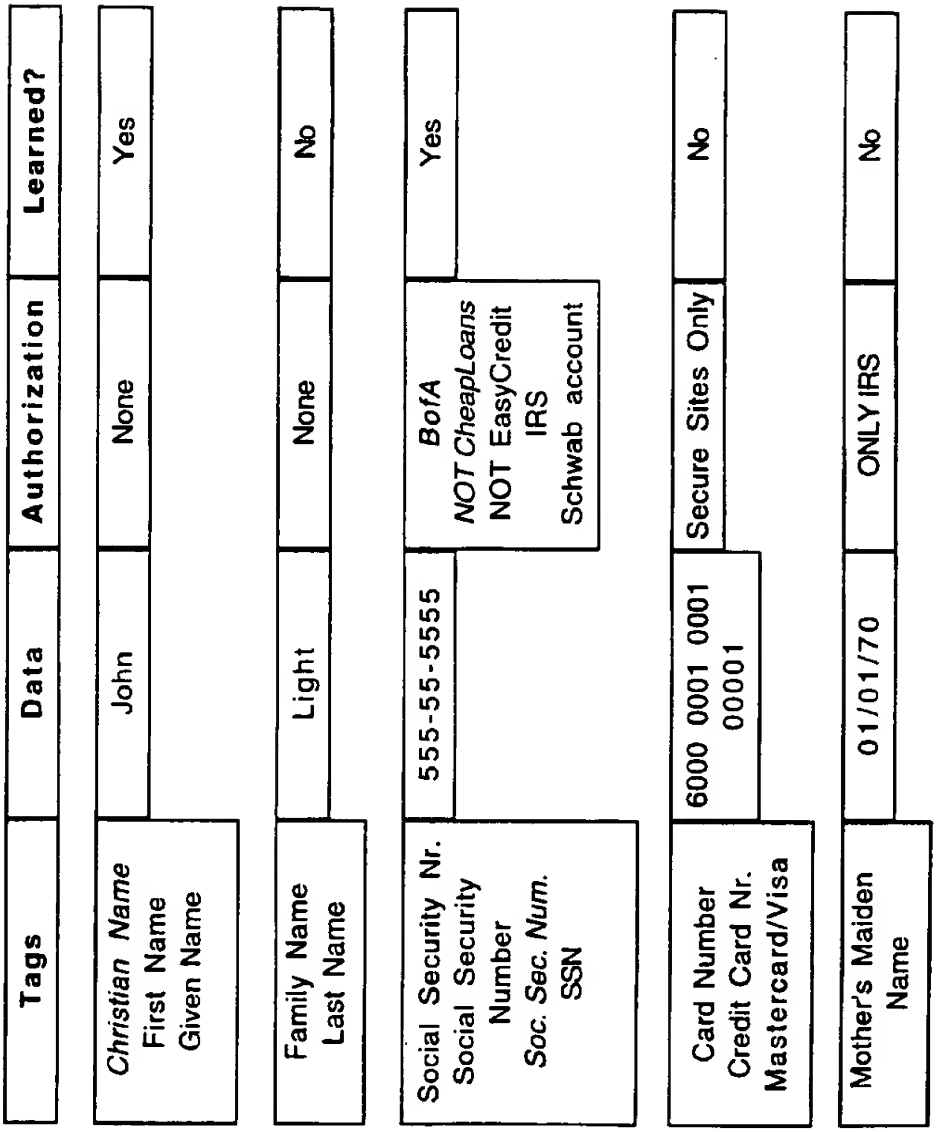

Tags
Data
Authorization
Learned?
Christian Name
First Name
Given Name
John
None
Yes
Family Name
Last Name
Light
None
No
Social Security Nr.
Social Security
Number
Soc. Sec. Num.
SSN
555-55-5555
BofA
NOT CheapLoans
NOT EasyCredit
IRS
Schwab account
Yes
Card Number
Credit Card Nr.
Mastercard/Visa
6000 0001 0001
00001
Secure Sites Only
No
Mother's Maiden
Name
01/01/70
ONLY IRS
No

Fig. 6

AUTOMATIC WEB BASED FORM FILL-IN

FIELD OF THE INVENTION

The present invention relates to , and more specifically, to

BACKGROUND

The World-Wide Web (WWW, W3, the Web) is an Internet client-server hypertext distributed information retrieval system. An extensive user community has developed on the Web since its public introduction. On the Web everything (documents, menus, indices) is represented to the user as a hypertext object in hypertext markup language (HTML) format. Hypertext links refer to other documents by their universal resource locators (URLs). The client program, known as a browser, e.g. NCSA Mosaic, Netscape Navigator, runs on the user's computer and provides two basic navigation operations: to follow a link or to send a query to a server.

A variety of client and server software is freely available. Most clients and servers support "forms" which allow the user to enter arbitrary text as well as selecting options from customizable menus and on/off switches. As more business is transacted on the Web, forms are proliferating. The forms may include forms for requesting further information, for ordering items from the Web, for registering for a Web site, etc.

Currently, the user has to fill out each of these forms separately. Generally, the forms request the same types of information, i.e. name, address, telephone number, e-mail address, etc. The user has to enter all of this information for each form. This is repetitious and takes time. Additionally, if such information as credit card number or social security number is requested, the user has to pull out the credit card and copy a long string of numbers. This makes errors likely. Furthermore, the user has to verify that a Web site that requests a credit card number or similar information generally kept confidential, is of the appropriate level of security for the user to feel comfortable sending the information over the Web.

SUMMARY OF THE INVENTION

A method for filling in forms in a web page is described. A web page is accessed. A form included in the web page is recognized. Data is automatically filled into the form from a database.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

FIG. **1** is one embodiment of a network on which the present invention may be implemented.

FIG. **2** is one embodiment of a computer system on which the present invention may be implemented.

FIG. **3** is a block diagram illustrating one embodiment of the present invention.

FIG. **4** is a flowchart illustrating one embodiment of the initial setup of the present invention.

FIG. **5A** is a flowchart illustrating one embodiment of the fill-in process.

FIG. **5B** is a flowchart illustrating the learning process associated with the fill-in process of FIG. **5A**.

FIG. **6** illustrates sample database entries.

DETAILED DESCRIPTION

A method and apparatus for automatic web form fill-in is described.

FIG. **1** is one embodiment of a network on which the present invention may be implemented. The user's system, a client, **110** is coupled to a network **120**. The client **110** may be coupled to the network **120** via a modem connection, an Ethernet connection, a local area network (LAN), a wide area network (WAN), or any other type of network connection. Servers **130** are coupled to the network **120**. For one embodiment, the server **130** may be the same computer as the client **110**. For one embodiment, these servers **130** provide Web pages to the user via the network **120**. These Web pages may include forms, as will be discussed below.

FIG. **2** is one embodiment of a computer system on which the present invention may be implemented. FIG. **2** is a block diagram of the computer system **200** in which an embodiment of the present invention can be implemented. Computer system **200** comprises a bus **201** or other communication means for communicating information, and a processor **202** coupled with bus **201** for processing information. Computer system **200** also comprises a read only memory (ROM) and/or other static storage device **204** coupled to bus **201** for storing static information and instructions for processor **202**.

The computer system **200** further comprises a main memory **203**, a dynamic storage device for storing information and instructions to be executed. Main memory **203** also may be used for storing temporary variables or other intermediate information during execution of instructions. In one embodiment the main memory **203** is dynamic random access memory (DRAM).

Computer system **200** can also be coupled to a display device **205**, such as a cathode ray tube (CRT) or liquid crystal display (LCD) screen, for displaying information to a computer user. An alphanumeric input device **206** is typically coupled to the computer system **200** for communicating information and command selections to processor **202**. The input device **206** may be a cursor control device **206**, such as a mouse, a trackball, trackpad, or cursor direction keys for communicating direction information and command selections to processor **202**, and for controlling cursor movement on display device **205**. Alternatively, other input devices **206** such as a stylus or pen can be used to interact with the display. Multiple input devices **206** may be coupled to the computer system **200**.

The computer system **200** may further be coupled to a network device **210**. The network device **210** may be a modem, an Ethernet link, or similar device for connecting the computer system **200** to a network.

FIG. **3** is a block diagram illustrating one embodiment of the present invention. For one embodiment, the present invention is part of a browser. A browser is a program which allows a person to read hypertext. The browser gives some means of viewing the contents of web pages (or nodes) and of navigating from one node to another. For an alternative embodiment, the present invention is not part of a browser, but rather an independent software unit, that interacts with the browser. The browser receives a web address from the user, and opens the corresponding web page.

The auto-fill-in system **310** includes a fill-in subunit **315** and a learning subunit **370**. The fill-in subunit **315** includes a form recognition unit **320**. When a form is included in the web page the form recognition unit **320** notes that there is a form. For one embodiment, the form includes an hypertext

markup language (HTML) tag such as "form", or "input type" indicating that it is a form or that it requires user input. The auto-fill-in system **310** then inspects the source code for the page, and recognizes tags associated with blank spaces in the form. For example, a form may look as follows:

We encourage you to enter your credit card number on-line, this is why it's secure. However, you also have the option of phoning us with the number.

Please enter your e-mail address: [ ]

My password is: [ ]

Have you forgotten your password?

My credit card type is: O MC O Visa O AmEx

My credit card number is: [ ]

The source code of the form may look as follows:

<form method=POST action=/exec/obidos/order-form-page1/6474-2122890-104042>

We encourage you to enter your credit card number online (<ahref="/exec/obidos/subst/help/payment.html/6474-2122890-104042#credit-cards"><fontsize="-1">why this is safe). However, you also have the option of phoning us with the number.

<blockquote>

Please enter your e-mail address:
<input type=text name=email size=40 value="">

My password is <input type="password" size="20"name="password" maxlength=20>.

Have you forgotten your password?<p>

Credit card type

<input type=radio name=creditcardtype=MC>

<input type=radio name=creditcardtype=Visa>

<input type=radio name=creditcardtype=AmEx>

My credit card number is <input type="cardnumber" size="16" name="cardnumber" maxlength=24>.

</blockquote>

The form recognition unit **320** recognizes tags such as "input type" that connote forms. The form recognition unit **320** then passes the entire source of the web page to the tag recognition unit.

The tag recognition unit **350** then scans the form, and determines what the form is asking for. Thus, for example, in this instance, the name of the first item is "email". Alternately, the tag recognition unit **350** may recognize the label displayed to the user for the specified entry. Thus, for example the text "please enter your e-mail address" may be recognized by the tag recognition unit **350**, and "e-mail address" extracted from it. For one embodiment, the displayed label or the "name" associated with the blank is the tag recognized by the tag recognition unit **350**. For one embodiment, the name associated with the blank is the preferred tag.

Once the tag recognition unit **350** has extracted a tag, it passes the tag to the matching unit **360**. The matching unit **360** searches in the database **390** for a similar tag. For one embodiment, the matching unit **360** has some intelligence, and corrects singulars v. plurals, misspellings, words that were combined into a single word, etc. Some of the entries in the database are illustrated in FIG. **6**. The matching unit **360** determines whether there is a tag that is "email" or "e-mail address". If the matching unit **360** finds a matching tag in the database **390**, it passes the tag, the data associated with the tag, and the authorization of the tag to the authorization evaluation unit **340**.

The authorization evaluation unit **340** determines whether there are any restrictions on the data. Such restrictions may include restricting the data to only specific sites, or only secure sites, and similar restrictions. The authorization evaluation unit **340** compares the web page with the authorization data associated with the information. If the web page is authorized to receive the data, the authorization evaluation unit **340** passes the data to the filling unit **330**. The filling unit **330** inserts the data into the space associated with the tag.

In this way, the spaces in the form are filled in. If, for example, there are blank spaces, the auto-fill-in system **310** waits for the user to fill in any blanks. When the user presses enter, or otherwise indicates that the form is completely filled in, the learning subunit **370** scans the form, and determines whether there are any spaces that were filled in by the user, not the fill-in subunit **315**. The learning subunit **370** then extracts the tags and data associated with these user-filled-in spaces, and passes them to the learning subunit **370**.

The learning subunit **370** determines whether the data already exists in the database **390**. If it does, the database adding unit **385** adds the new tag to the list of tags associated with the information in the database **390**. If the data is not in the database **390**, the database adding unit **385** adds the new data and the new tag to the database **390**.

FIG. **4** is a flowchart illustrating one embodiment of the initial setup of the present invention. Generally, the user will wish to initially enter the personal information to be filled into the various forms. Alternatively, this step may be skipped, and the system may only learn from user input, as will be described below.

At block **410**, the initial setup starts. At block **420**, the existing list of tags is displayed. For example, this list of tags may include "First name", "Last Name", "e-mail address", etc. For one embodiment, this list of tags may be included with the application. Alternatively, the user may be questioned for tags initially.

At block **430**, the user is prompted to enter appropriate data for the existing tags. This may include information such as a name, e-mail address, credit card numbers, social security number, etc.

At block **440**, the user is requested to enter further tags associated with the data. Thus, for example, when the user enters his or her first name, in response to a tag asking for a "first name", the user may add other tags, such as "given name", etc.

At block **450**, the user is requested to enter the authorization level for the data. Data may be divided into multiple categories, as illustrated for example in FIG. **6**. Data may have no authorization restrictions. Information such as name and e-mail address may be generally released to all sites that ask for them.

Alternately, data may be restricted to only a certain one or list of sites. Thus, for example, for a social security number, the user may enter that the social security number may be released to the IRS, to the user's bank, etc. The user may further specify locations to which the information should not be released. Thus, for example, if there is page that is regularly visited that the user does not wish to release the data to, negative authorizations may also be entered.

A second type of authorization includes exclusive authorization. This is illustrated in FIG. **6** as well. The entry

tagged "mother's maiden name" which is often used by credit card companies for identification, may be restricted to be released only to the IRS. An authorization restricted as exclusive may include a list of one or more locations to which the data may be provided. When the user encounters a form that asks for data restricted by exclusive authorization, i.e. a page that asks for the user's mother's maiden name, the system does not query whether the user wishes to fill in the information. Rather, if the site is not in the list of sites, the system does not fill in the information, and assumes that the user will not release the information.

A third type of authorization is "secure site" authorization. Secure site authorization may include sites that have a verified certification from a recognizes certification authority, this may include encrypted sites, or otherwise secured sites. The security level may be set by the user. For one embodiment, all sites running secure hypertext transmission protocol (https) or a secure sockets layer (SSL) are deemed secure sites. Alternative authorization levels may be included, or may be defined by the user.

At block **460**, the system tests whether there are any blank tags remaining. The user may indicate that he or she does not wish to enter data for a preexisting tag. In that instance, the data associated with that tag is set to null, but not considered a blank tag for the purposes of the preliminary entry of data.

If there are blank tags, the system loops back to block **420**, and displays the tag list that has not been completed. If there are no blank tags, the system continues to block **470**.

At block **470**, the user is prompted to add additional tags. The user may, for instance, often use a system that requires age information. Thus, the user may add "age" as a tag, and fill in his or her age as data. At block **480**, the system tests whether more tags have been added. If more tags were added, the system returns to block **420**, and displays the added tags to the user for authorization level, etc. If no more tags were added by the user at block **470**, the preliminary data gathering is ended, and the flowchart stops at block **490**.

FIG. **5A** is a flowchart illustrating one embodiment of the fill-in process. The process starts at block **510**. At block **515**, the process tests whether a form has been encountered. If no form has been encountered, the process returns to block **515**. For one embodiment, this process is activated every time a new web page is opened. For one embodiment, the process runs in the background. If no forms were found at block **515**, the process returns to the background state, at block **510**. If a form is found, the process continues to block **520**. Alternatively, the fill-in process may be activated by the user. For one embodiment, the user may activate the fill-in process by pressing a key, a key combination, a left mouse button, or a similar activation mechanism.

At block **520**, a tag is copied. Each form has at least one entry blank to be filled in by the user. A least one tag is associated with every entry blank, indicating what the user should enter into the form. For one embodiment, the name of the input is copied as a tag. Thus, in the example above, the name "cardnumber" may be copied as a tag. For another embodiment, a displayed label associated with the entry blank may be copied. Thus, the text "My credit card number is:" is copied, and the tag "credit card number" is extracted from the text. For another embodiment, both the displayed label and the name are extracted as tags.

At block **525**, the tag is matched to a list of tags in the database. The database includes all of the tags originally supplied, tags entered by the user, and tags learned, as will be discussed later. The extracted tag is compared to the tags in the database.

At block **530**, the process tests whether there was a match between the extracted tag and the list of tags in the database. If there is no match, the process goes to block **570**. At block **570**, the process tests whether there are any more tags to check. If there are tags remaining to be checked, the process returns to block **520**. If there are no remaining tags to be checked, the process continues to block **575**, and the process waits for the user to send the form. If, at block **530**, a match was found, the process continues to block **535**.

At block **535**, the process tests whether the data associated with the matching tag found in the database is available for this web page. As discussed above, there are various levels of authorization for data. Thus, at block **535**, the process tests whether the data is authorized to be released to the web page in question.

At block **540**, the process determines whether or not the data is available. If the data is available, the process continues to block **545**. At block **545**, the data is filled into the form. The process then continues to block **570**, where it tests whether there are any more tags to check.

If, at block **540**, it is determined that the data is not available, the process continues to block **550**. For one embodiment, the process collects all of the data that is not properly authorized, and tests authorization for all of the data at the same time. In other words, only after no blank spaces remain does the process continue to block **550**.

At block **550**, the user is queried whether it is acceptable to fill-in the data. For one embodiment, the user is queried only if the authorization level is not set to exclusive authorization. For another embodiment, the user is not queried if the page is on the exclusion list, as described above.

At block **555**, it is tested whether it is acceptable to fill-in the data. If it is acceptable to fill-in the data, the process continues to block **560**. At block **560**, the web page is added to the list of authorized sites for which the data is available. The process then continue to block **545**.

At block **555**, if it is determined that it is not acceptable to enter the data, the process continues to block **565**. At block **565**, the web page is added as a negative authorization That is, if this web site is encountered in the future, the user is not queried whether the data is available, but rather, the blanks are automatically left blank. From block **565**, the process returns to block **570**, and queries whether there are more tags to check.

FIG. **5B** is a flowchart illustrating the learning process associated with the fill-in process of FIG. **5A**. In FIG. **5A**, the blanks are automatically filled by the system. When the last blank is filled in, the system waits for the user to send the form, at block **575**. However, the user may enter additional data prior to sending the form. FIG. **5B** illustrates the process occurring concurrently with, or after, waiting for the user to send the form, at block **575** of FIG. **5A**.

Returning to FIG. **5B**, at block **580**, the system tests if there are any blanks, i.e. areas which the automatic fill-in did not complete. If there are no blanks, the learning process is finished. If there are blanks, the system, at block **585**, tests whether the user filled in any of the blanks. In many forms, areas may be left blank. Thus, the user may not chose to complete every entry on the form. If, at block **585**, the process finds that the user did not fill in any blanks, the learning process is finished. If the user filled in at least one blank, the process continues to block **590**.

At block **590**, the system queries whether the data filled in matches information in the database. This is applicable if a different tag is used by the web page for known data. For example, the tag "Christian name" may be used in a foreign web page, for the data tagged "first name" in the database. The data entered by the user would still be "John", or the appropriate first name.

If the data matches information in the database, the process continues to block **595**. At block **595**, the new tag is added to the list of tags associated with the information found in the database. Thus, the tag "Christian name" would be added to the tags associated with the data "John" in the above example. The process then returns to block **585**, to query whether any other blanks were filled in by the user.

If, at block **590**, it is found that the data does not match information in the database, the process continues to block **597**.

At block **597**, the user is queried whether the new data should be added to the database. If the user replies in the negative, the process returns to block **585**, and the system again queries whether any other blanks were filled in by the user. If the user replies in the affirmative, the process continues to block **595**. At block **595**, the new tag and new data associated with it are added to the database. For another embodiment, the tag and data are automatically added to the database.

FIG. **6** illustrates sample database entries, as discussed above. Other data may of course be included in the database. Alternative arrangements of data may include not having an authorization, not having an indicator whether anything in the list was learned, etc.

In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

What is claimed is:

1. A method comprising:

recognizing a form in a web page;

identifying information to be filled into the form;

determining whether data corresponding to the information to be filled into the form is authorized by a user to be disclosed to the web page;

automatically filling the data into the form from a database if the data is authorized by the user to be disclosed to the web page.

2. The method of claim **1**, wherein recognizing the form further comprises extracting tags from the web page.

3. The method of claim **2**, wherein recognizing the form further comprises:

comparing the tags with a stored tag list in the database;

identifying a matched tag; and

inserting the data corresponding to the matched tag into the form.

4. The method of claim **3**, wherein determining whether the data corresponding to the information to be filled into the form is authorized to be disclosed to the web page comprises:

determining an authorization of the data; and

comparing the authorization of the data with an authorization level of the web page.

5. The method of claim **4**, further comprising:

if the web page is not authorized for the data, prompting the user to decide whether the web page should be authorized for the data; and

if the user decides that the web page should be authorized for the data, inserting the data and adding the web page to a list of authorized web pages for the data.

6. The method of claim **5** further comprising, if the user decides that the web page should not be authorized for the data, adding the web page to a list of not authorized web pages for the data.

7. The method of claim **1**, further comprising:

determining if the user filled additional data into blank fields in the form;

if the user did fill additional data into blank fields in the form, determining if the additional data corresponds to data already stored in the database; and

if the additional data corresponds to data already stored in the database, adding a tag associated with the additional data to a list of tags associated with the data already stored in the database.

8. The method of claim **7**, further comprising:

determining if the list of tags has an authorization list; and

if the list of tags has an authorization list, adding the web page on which the blank field was found to the authorization list for the data already stored in the database.

9. The method of claim **7**, further comprising:

determining if the additional data does not correspond to the stored data;

if the additional data does not correspond to the stored data, storing the additional data and the tag associated with the additional data in the database.

10. The method of claim **9**, further comprising automatically authorizing the additional data for the web page on which the blank fields were found.

11. The method of claim **9**, further comprising prompting the user to enter a security level for the additional data entered into the blank fields.

12. A method comprising:

opening a web page;

recognizing a form in the web page;

extracting tags from the form in the web page;

comparing the tags with a stored tag list in the database;

identifying a matched tag in the database;

determining whether the web page is authorized for the data corresponding to the matched tag;

if the web page is authorized for the data, inserting the data into the form in the web page; and

if the web page is not authorized for the data:

prompting a user to decide whether the web page should be authorized for the data;

if the user decides that the web page should be authorized for the data:

inserting the data into the form; and

adding the web page to a list of authorized web pages for the data; and

if the user decides that the web page should not be authorized for the data, adding the web page to a list of unauthorized web pages for the data.

13. A system comprising:

a plurality of personal data, tags, and an authorization level associated with the personal data;

a form recognition unit for recognizing information requested by a form in a web page;

an authorization evaluation unit for determining the authorization level of the personal data corresponding to the information requested by the form, and for determining an authorization level of the web page; and

a fill-in unit for filling the personal data from the database into the form, if the authorization evaluation unit authorizes the personal data for the web page.

14. The system of claim **13**, further comprising a tag extraction logic for extracting tags from the form in the web page.

15. The system of claim 15, further comprising:

a matching unit for comparing the tags extracted from the form with a stored tag list in the database and identifying a matched tag; and

wherein said fill-in unit receives the personal data from the matching unit if the matched tag is found.

16. The system of claim 16, wherein the authorization evaluation unit authorizes the matching unit to pass the personal data to the fill-in unit if the web page is authorized for the personal data.

17. The system of claim 17, wherein the authorization evaluation unit determines whether the web page should be authorized for the data and, if the web page should be authorized for the data, inserts the data and adds the web page to a list of authorized web pages for the data.

18. The system of claim 13, further comprising:

a learning subunit for adding personal data to the database, the personal data being entered by a user and not having been previously included in the database.

19. A method comprising:

opening a web page;

recognizing a form in a web page;

extracting tags from the web page;

comparing the tags with a stored tag list in a database;

identifying a matched tag;

determining whether data corresponding to the matched tag is authorized to be disclosed to non-listed sites;

prompting a user to decide if the web page should be authorized for the data, if the data is not authorized to be disclosed to non-listed sites; and

inserting the data corresponding to the matched tag into the form, if the user decides that the web page should be authorized for the data, or if the data is authorized to be disclosed to non-listed sites.

* * * * *

US006490601B1

(12) **United States Patent**
Markus et al.

(10) **Patent No.: US 6,490,601 B1**
(45) **Date of Patent: Dec. 3, 2002**

(54) **SERVER FOR ENABLING THE AUTOMATIC INSERTION OF DATA INTO ELECTRONIC FORMS ON A USER COMPUTER**

(75) Inventors: **Matthew A. Markus**, San Francisco, CA (US); **Erick M. Herrarte**, Berkeley, CA (US)

(73) Assignee: **Infospace, Inc.**, Bellevue, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/231,644**

(22) Filed: **Jan. 15, 1999**

(51) **Int. Cl.**[7] **G06F 17/21**
(52) **U.S. Cl.** **707/507**; 705/80; 709/219
(58) **Field of Search** 707/507–508, 707/100, 505, 513; 705/35, 37, 80; 709/217–219

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,537,586	A *	7/1996	Amram et al.	395/600
5,987,440	A *	11/1999	O'Neil et al.	705/39
6,192,380	B1 *	2/2001	Light et al.	707/10
6,199,079	B1 *	6/2001	Gupta et al.	707/507
6,327,598	B1 *	12/2001	Kelley et al.	707/501.1
6,345,278	B1 *	2/2002	Hitchcock et al.	707/100
2001/0011250	A1 *	8/2001	Paltenghe et al.	705/41

FOREIGN PATENT DOCUMENTS

WO	WO 98/04976	2/1998

OTHER PUBLICATIONS

US 6,047,266, 4/2000, Bartoli et al. (withdrawn)*
Lemay, Laura. Java 1.1: Interactive Course, 1997, The Waite Group.*
Lemay, Laura. Java 1.1: Interactive Course, 1997, The Waite Group, pp. 171–174.*
Cozzens, Lisa. JavaScript Tutorial, http://www.cs.brown.edu/courses/bridge/1998/res/javascript-tutorial.html, 1998.*
Printout of Website for *About eWallet*, http://www.ewallet.com, Jan. 11, 1999, 4 pages.
Printout of Website for *Transactor Networks (CitiWallet)*, http://www.transactor.net.com, Jan. 11, 1999, 4 pages.
Bowen, Barry D., "How popular sites use cookie technology," *Netscape Enterprise Developer*, Apr. 1997.
Sirbu, Marvin , and J.D. Tygar, "NetBill: An Internet Commerce System Optimized for Network–Delivered Services," *IEEE Personal Communications*, 34–39, Aug. 1995.
Unknown: "Gator offers one–click shopping at over 5,000 e–commerce sites today," *Internet Publication*, Origin Unknown, Jun. 14, 1999.
Unknown, "E–Commerce Leaders Announce Universal Formats for Simplified Online Payments," *Internet Publication*, Origin Unknown, Jun. 14, 1999.

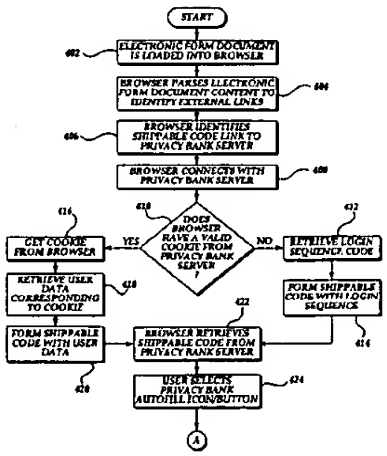
* cited by examiner

Primary Examiner—Joseph H. Feild

(57) **ABSTRACT**

Apparatus, methods, and computer program products are disclosed for constructing and transmitting an executable software module on a personal information server to a remote computer. The software module is constructed such that once received by a browser displaying a form, it is executed and user data is automatically inserted into an electronic form. The software module contains field names from a downloaded form and matching data items which are inserted into the form on the remote user computer. A method for constructing a shippable software module on a personal information server suitable for execution on a remote computer for inserting data strings into an electronic form is described. A form mapping containing a set of associations between fields in the electronic form ("non-standard fields") and pre-named fields ("standard fields") on the personal information server is retrieved. Each mapping is associated with a registered electronic form. A raw data file containing data strings, each data string corresponding to a pre-named field is retrieved. Each raw data file is associated with a registered user. The form mapping is utilized to attach a data string to the field in the electronic form where the pre-named field and the field in the electronic form have been previously matched or mapped.

22 Claims, 15 Drawing Sheets



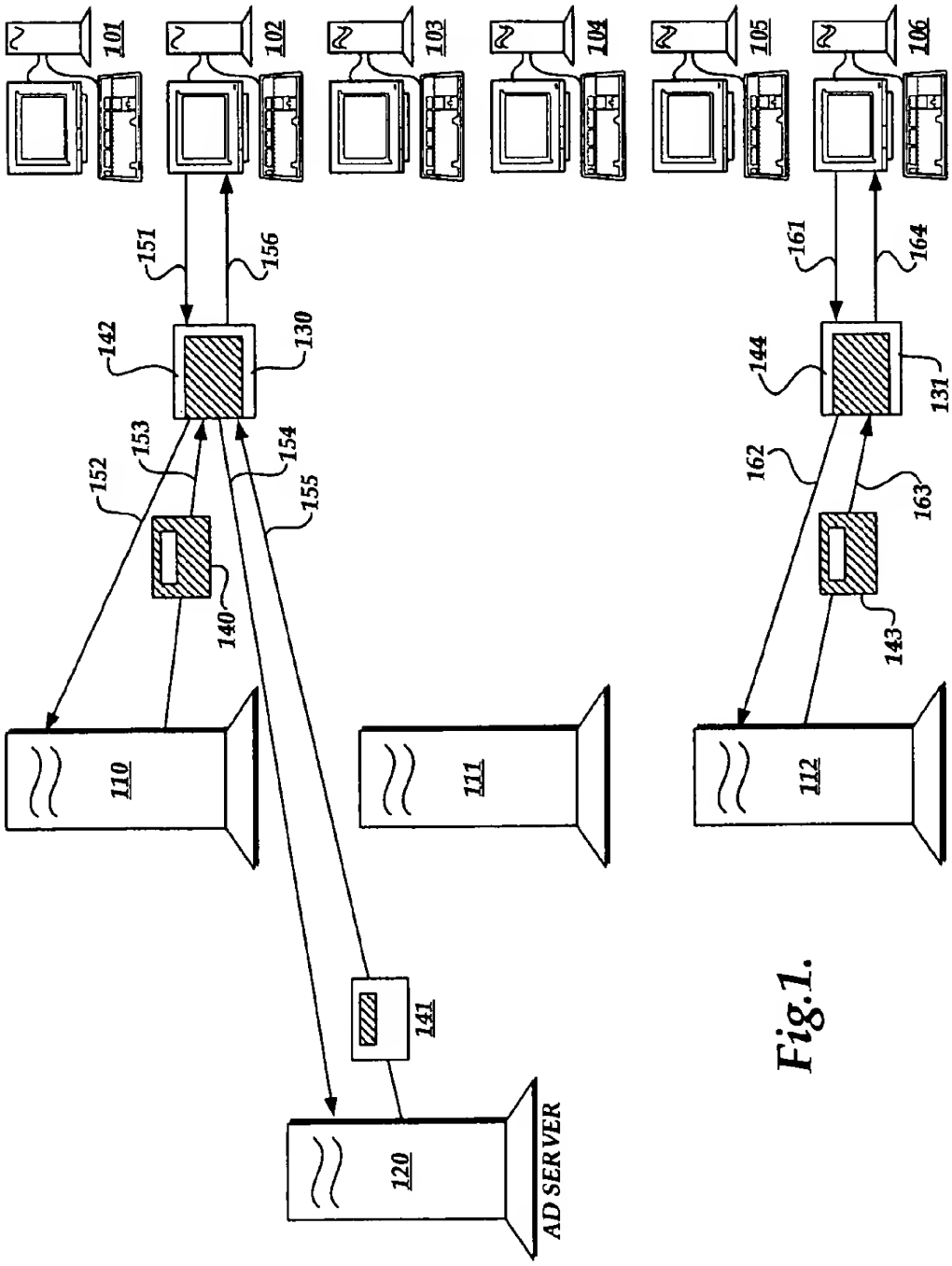

101
102
103
104
105
106
151
156
161
164
142
130
144
131
153
154
152
155
162
163
140
143
110
111
112
141
120
AD SERVER

Fig.1.

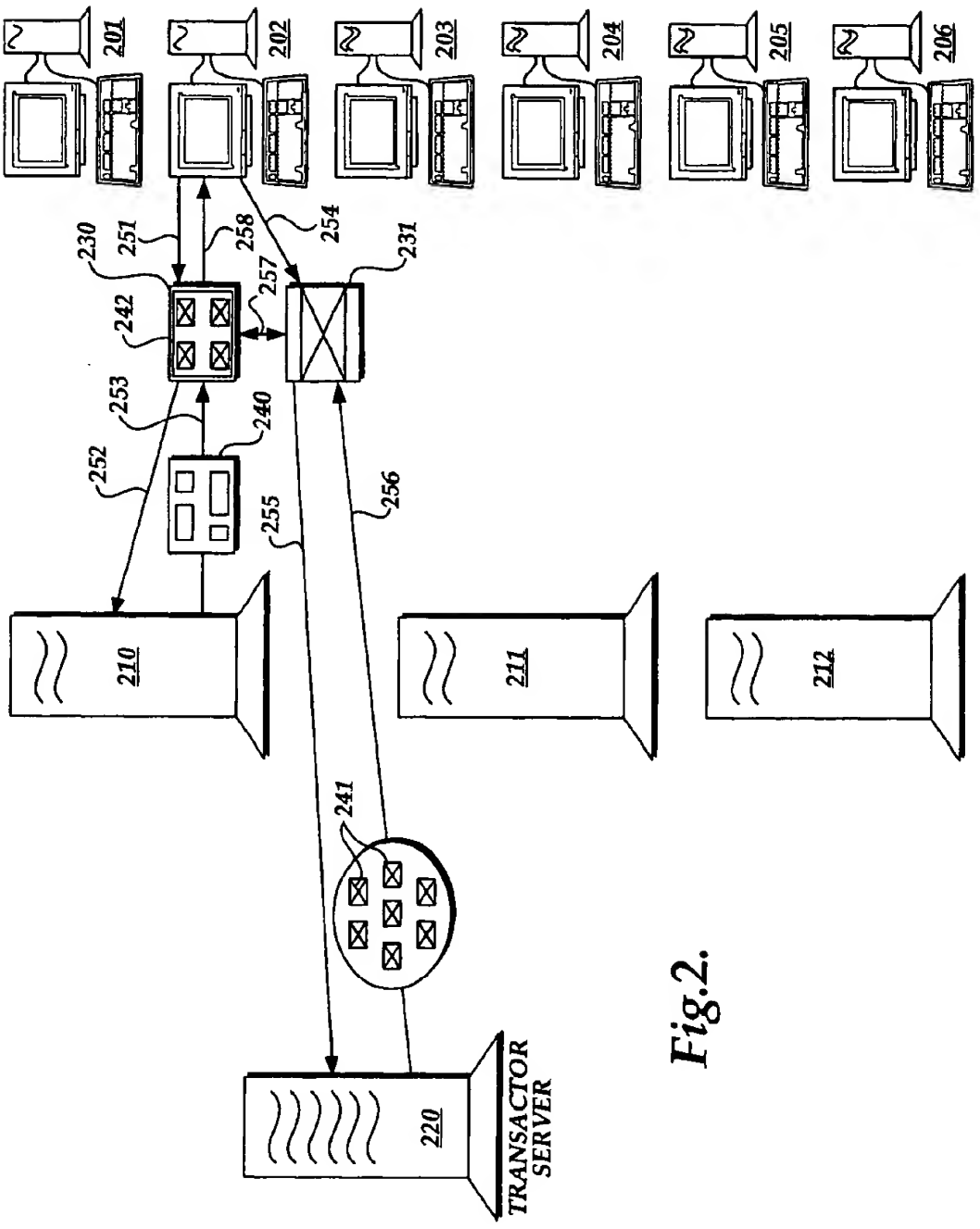

201
202
203
204
205
206
230
251
258
254
231
257
242
253
240
252
255
256
210
211
212
241
220
TRANSACTOR SERVER

Fig.2.

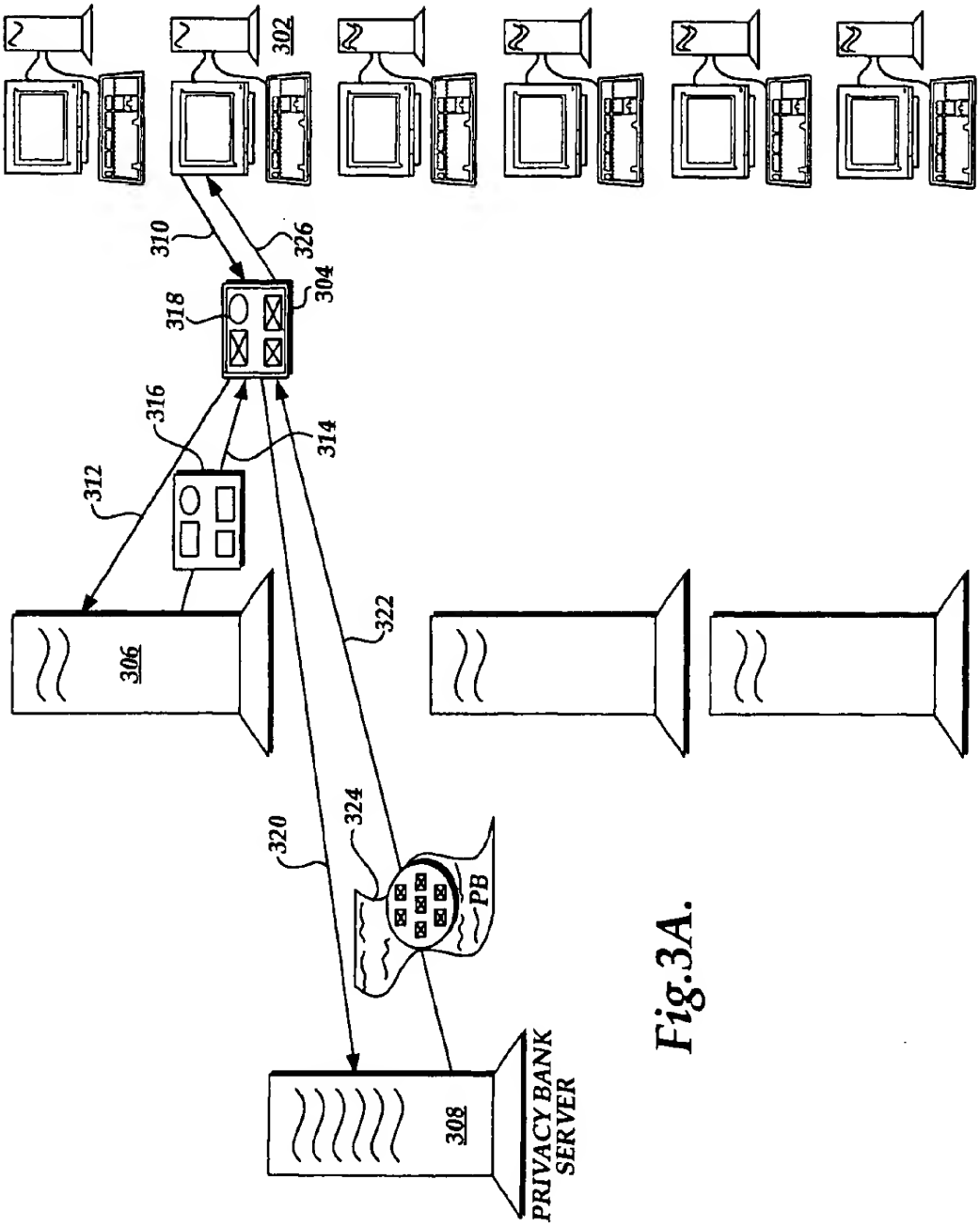

302
310
326
304
318
316
314
312
306
322
320
324
PB
308
PRIVACY BANK SERVER

Fig.3A.

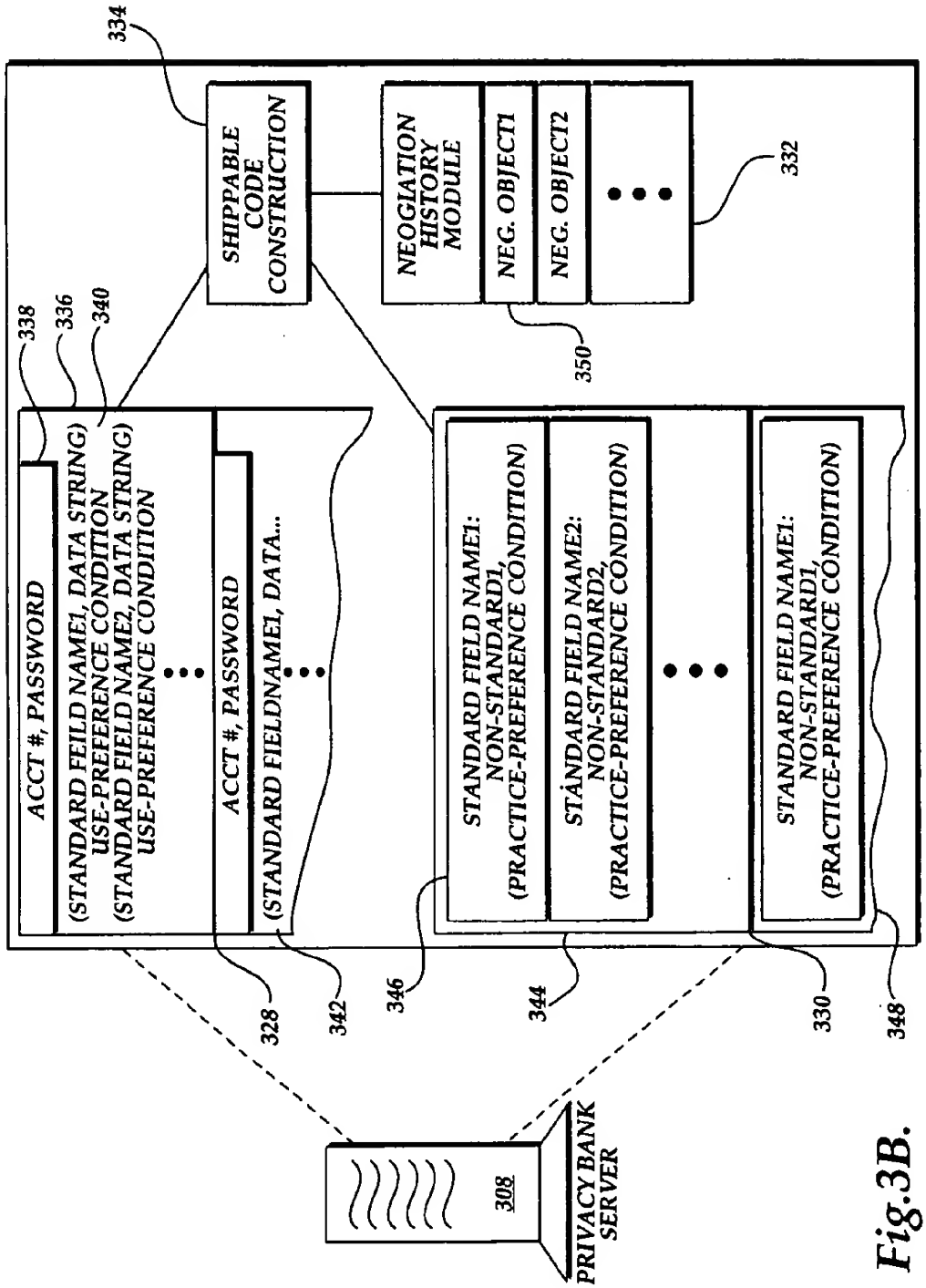

334
SHIPPABLE CODE CONSTRUCTION
NEOGIATION HISTORY MODULE
NEG. OBJECT1
NEG. OBJECT2
332
350
338
336
340
ACCT #, PASSWORD
(STANDARD FEILD NAME1, DATA STRING) USE-PREFERENCE CONDITION
(STANDARD FIELD NAME2, DATA STRING) USE-PREFERENCE CONDITION
ACCT #, PASSWORD
(STANDARD FIELDNAME1, DATA...
STANDARD FIELD NAME1: NON-STANDARD1, (PRACTICE-PREFERENCE CONDITION)
STANDARD FIELD NAME2: NON-STANDARD2, (PRACTICE-PREFERENCE CONDITION)
STANDARD FIELD NAME1: NON-STANDARD1, (PRACTICE-PREFERENCE CONDITION)
328
342
346
344
330
348
308
PRIVACY BANK SERVER

Fig.3B.

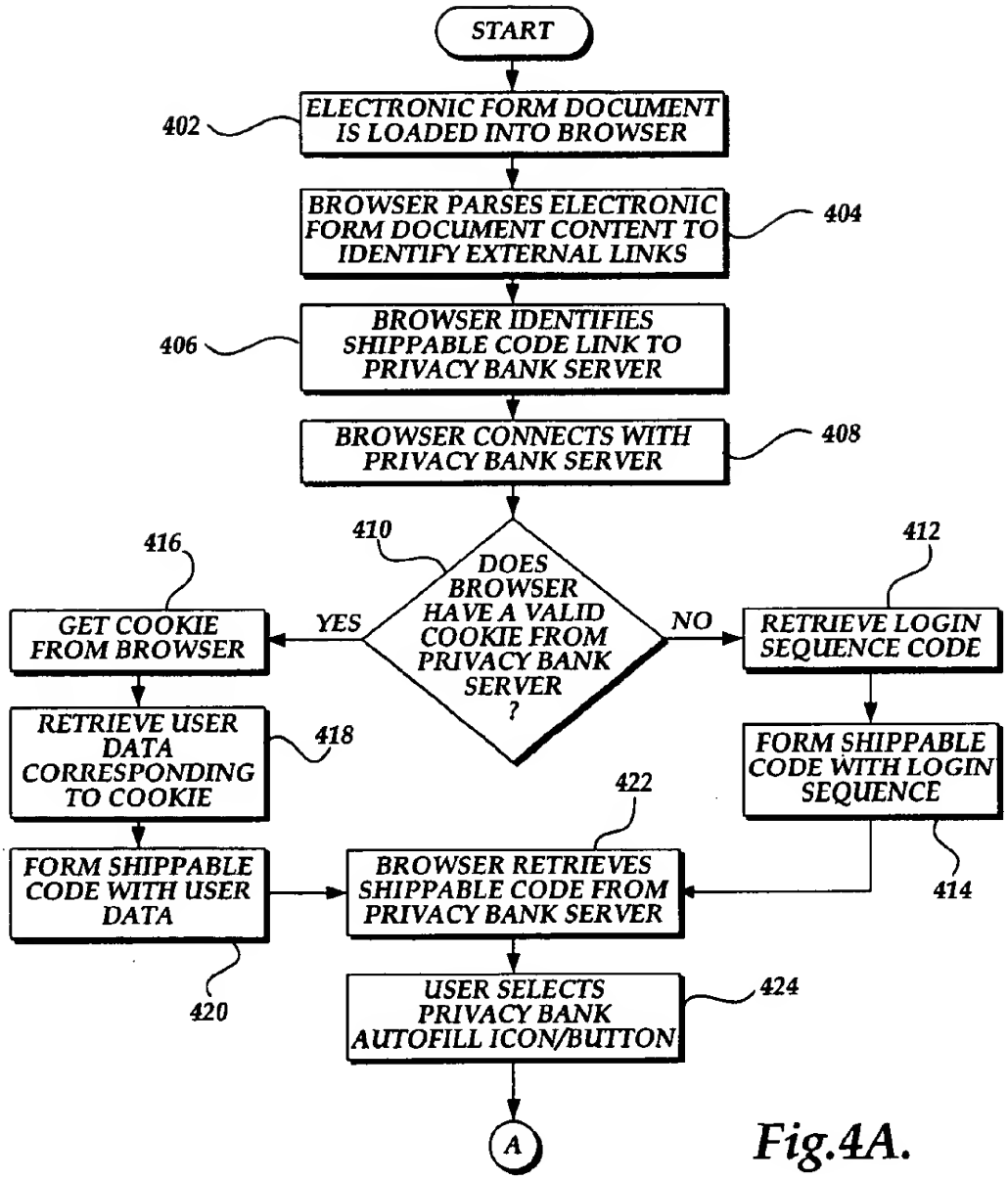

START
ELECTRONIC FORM DOCUMENT IS LOADED INTO BROWSER
402
BROWSER PARSES ELECTRONIC FORM DOCUMENT CONTENT TO IDENTIFY EXTERNAL LINKS
404
BROWSER IDENTIFIES SHIPPABLE CODE LINK TO PRIVACY BANK SERVER
406
BROWSER CONNECTS WITH PRIVACY BANK SERVER
408
410
DOES BROWSER HAVE A VALID COOKIE FROM PRIVACY BANK SERVER ?
YES
NO
416
GET COOKIE FROM BROWSER
412
RETRIEVE LOGIN SEQUENCE CODE
RETRIEVE USER DATA CORRESPONDING TO COOKIE
418
FORM SHIPPABLE CODE WITH LOGIN SEQUENCE
422
FORM SHIPPABLE CODE WITH USER DATA
BROWSER RETRIEVES SHIPPABLE CODE FROM PRIVACY BANK SERVER
414
420
USER SELECTS PRIVACY BANK AUTOFILL ICON/BUTTON
424
A

Fig.4A.

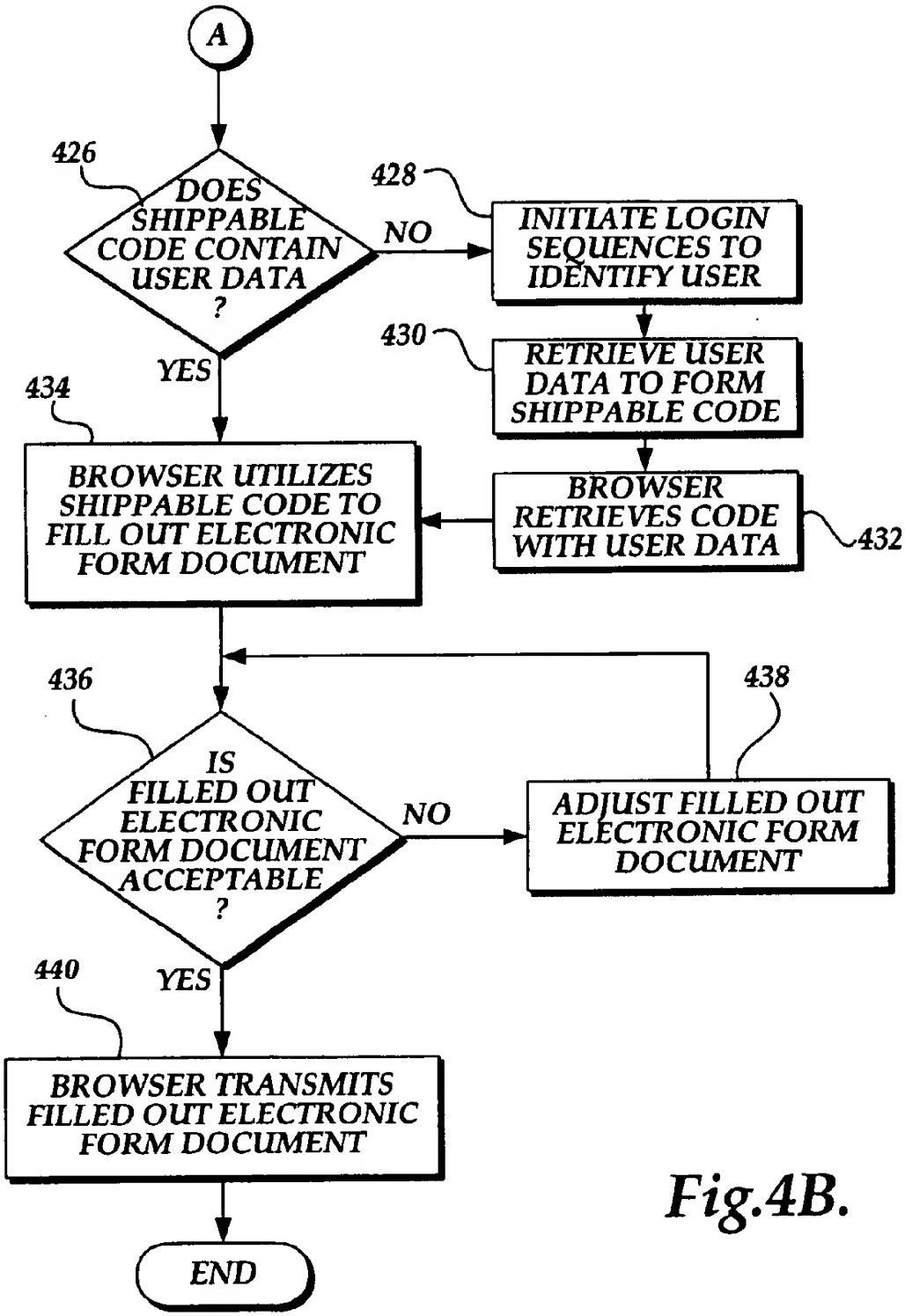

A
426
DOES SHIPPABLE CODE CONTAIN USER DATA ?
NO
428
INITIATE LOGIN SEQUENCES TO IDENTIFY USER
430
RETRIEVE USER DATA TO FORM SHIPPABLE CODE
YES
434
BROWSER UTILIZES SHIPPABLE CODE TO FILL OUT ELECTRONIC FORM DOCUMENT
BROWSER RETRIEVES CODE WITH USER DATA
432
436
438
IS FILLED OUT ELECTRONIC FORM DOCUMENT ACCEPTABLE ?
NO
ADJUST FILLED OUT ELECTRONIC FORM DOCUMENT
440
YES
BROWSER TRANSMITS FILLED OUT ELECTRONIC FORM DOCUMENT
END

Fig.4B.

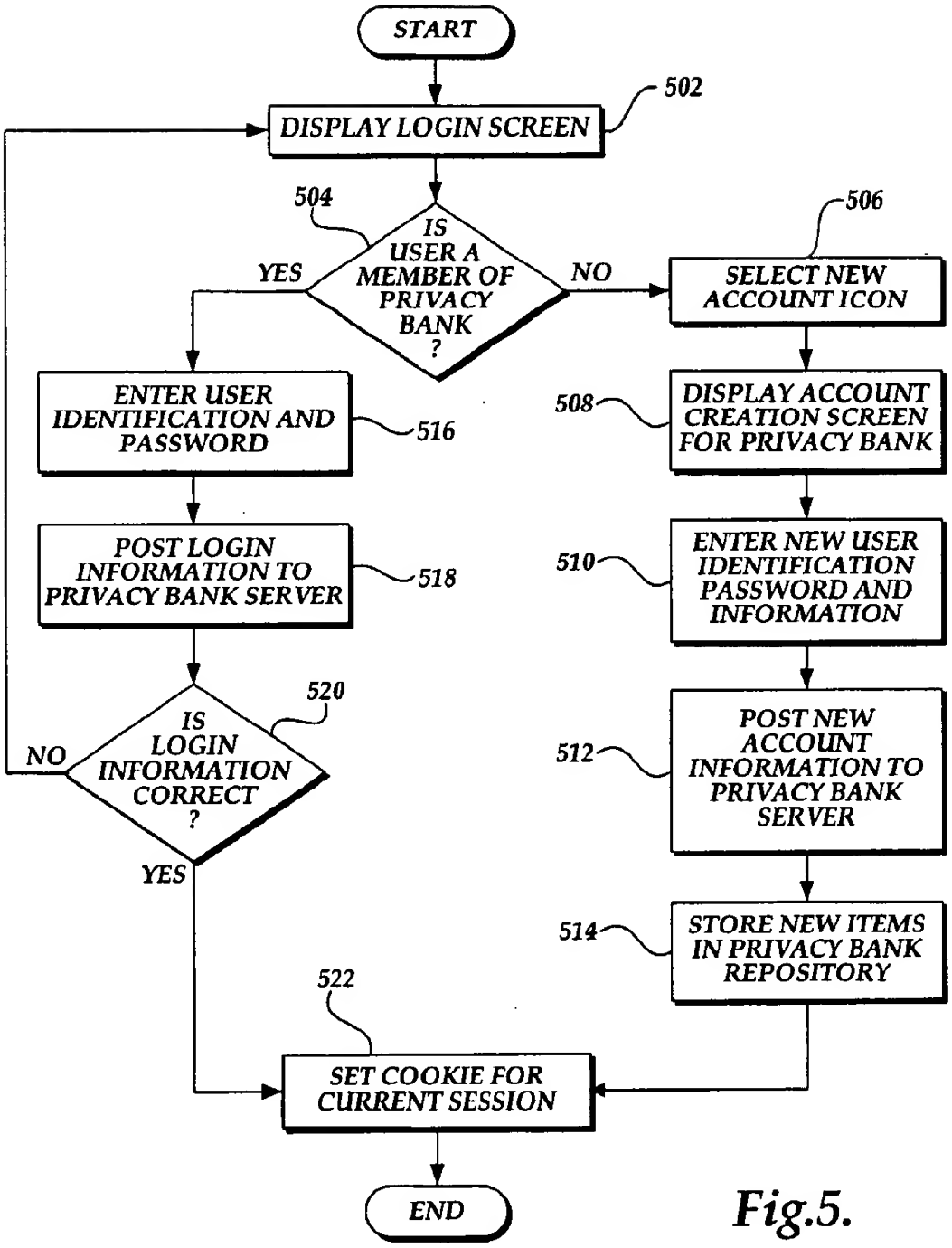

START
DISPLAY LOGIN SCREEN
502
504
IS USER A MEMBER OF PRIVACY BANK ?
YES
NO
506
SELECT NEW ACCOUNT ICON
ENTER USER IDENTIFICATION AND PASSWORD
516
508
DISPLAY ACCOUNT CREATION SCREEN FOR PRIVACY BANK
POST LOGIN INFORMATION TO PRIVACY BANK SERVER
518
510
ENTER NEW USER IDENTIFICATION PASSWORD AND INFORMATION
520
IS LOGIN INFORMATION CORRECT ?
NO
YES
512
POST NEW ACCOUNT INFORMATION TO PRIVACY BANK SERVER
514
STORE NEW ITEMS IN PRIVACY BANK REPOSITORY
522
SET COOKIE FOR CURRENT SESSION
END

Fig.5.

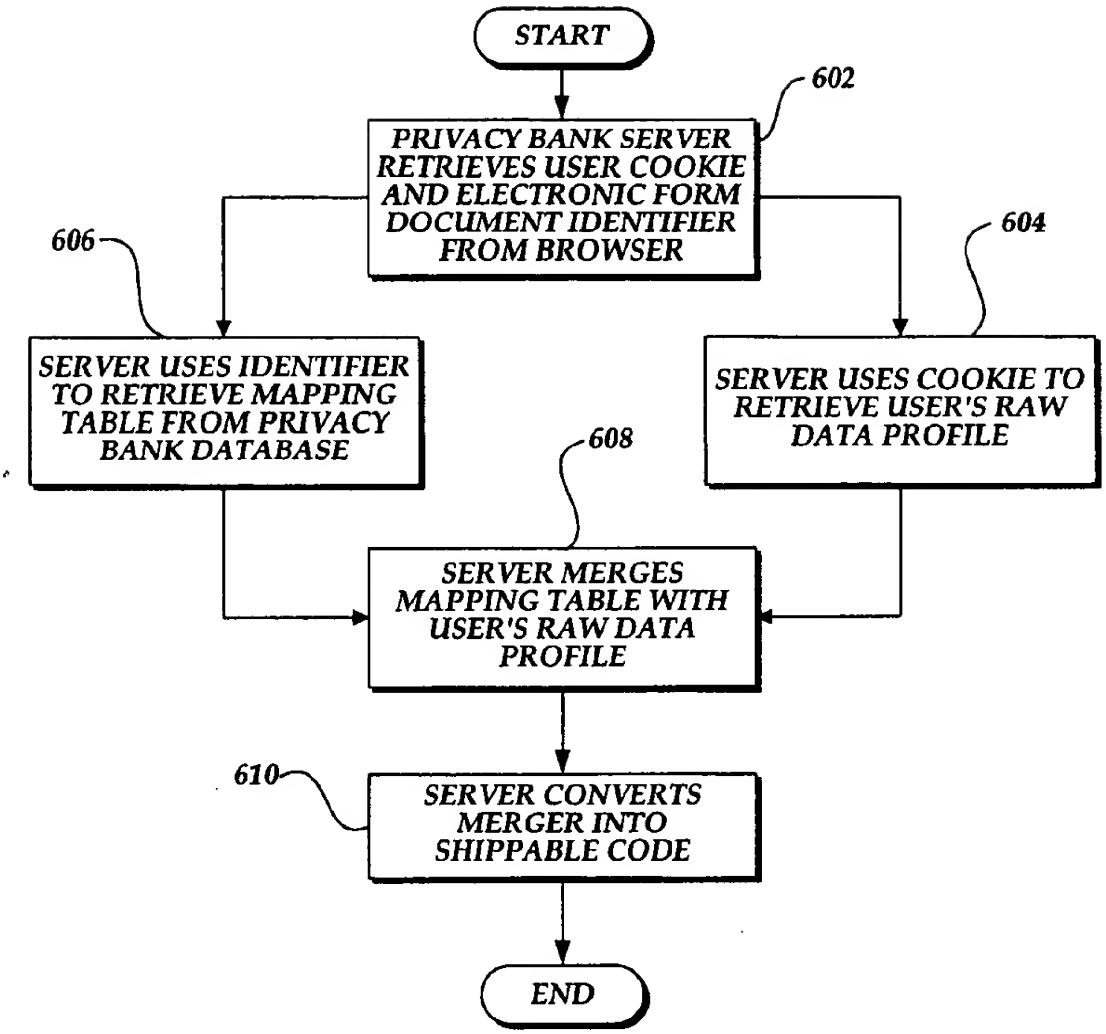

START
602
PRIVACY BANK SERVER RETRIEVES USER COOKIE AND ELECTRONIC FORM DOCUMENT IDENTIFIER FROM BROWSER
606
604
SERVER USES IDENTIFIER TO RETRIEVE MAPPING TABLE FROM PRIVACY BANK DATABASE
SERVER USES COOKIE TO RETRIEVE USER'S RAW DATA PROFILE
608
SERVER MERGES MAPPING TABLE WITH USER'S RAW DATA PROFILE
610
SERVER CONVERTS MERGER INTO SHIPPABLE CODE
END

Fig.6.

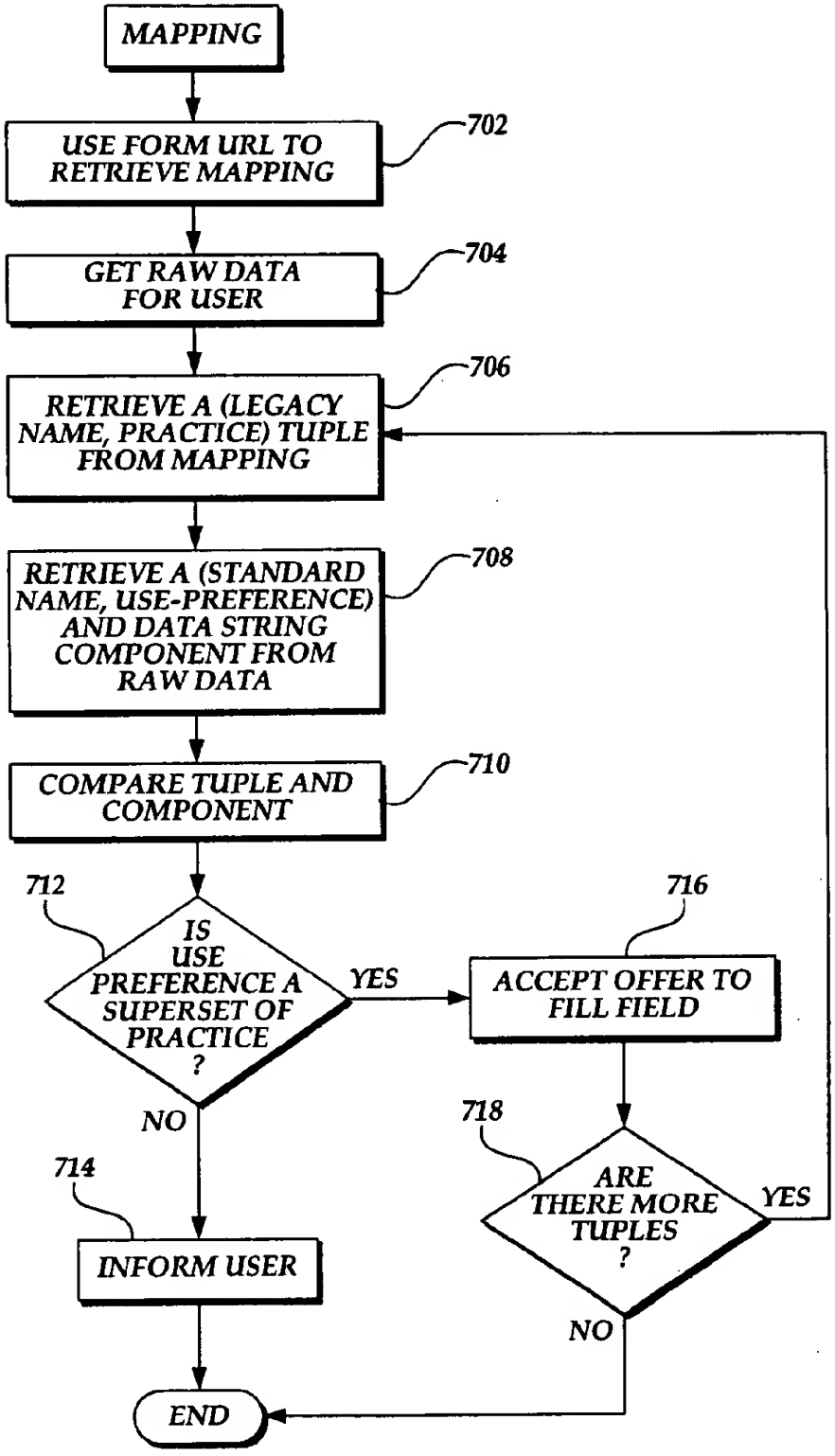

MAPPING
USE FORM URL TO RETRIEVE MAPPING
702
GET RAW DATA FOR USER
704
RETRIEVE A (LEGACY NAME, PRACTICE) TUPLE FROM MAPPING
706
RETRIEVE A (STANDARD NAME, USE-PREFERENCE) AND DATA STRING COMPONENT FROM RAW DATA
708
COMPARE TUPLE AND COMPONENT
710
712
IS USE PREFERENCE A SUPERSET OF PRACTICE ?
YES
716
ACCEPT OFFER TO FILL FIELD
NO
718
ARE THERE MORE TUPLES ?
YES
714
INFORM USER
NO
END

Fig.7.

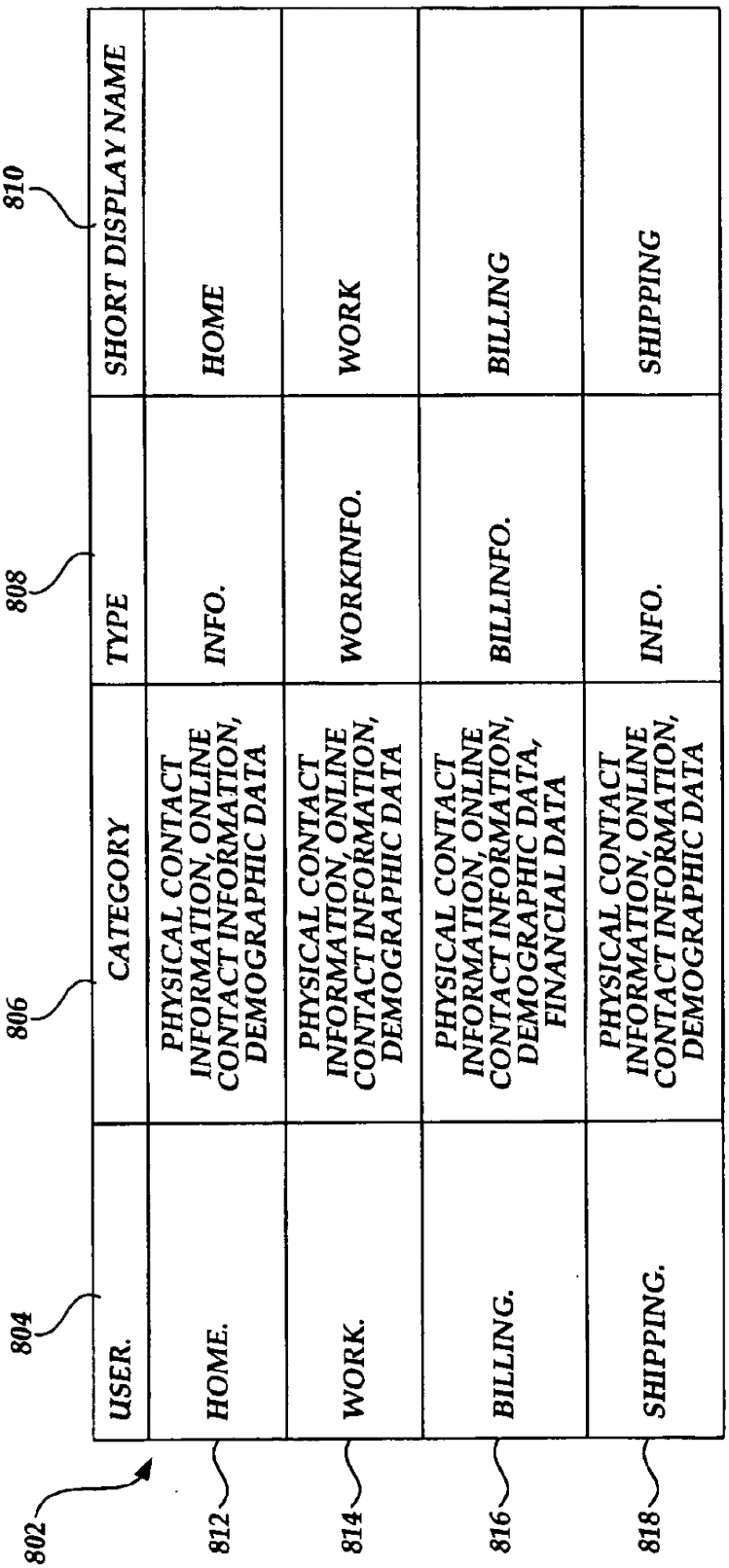

802
804
806
808
810
812
814
816
818
USER.
CATEGORY
TYPE
SHORT DISPLAY NAME
HOME.
PHYSICAL CONTACT INFORMATION, ONLINE CONTACT INFORMATION, DEMOGRAPHIC DATA
INFO.
HOME
WORK.
PHYSICAL CONTACT INFORMATION, ONLINE CONTACT INFORMATION, DEMOGRAPHIC DATA
WORKINFO.
WORK
BILLING.
PHYSICAL CONTACT INFORMATION, ONLINE CONTACT INFORMATION, DEMOGRAPHIC DATA, FINANCIAL DATA
BILLINFO.
BILLING
SHIPPING.
PHYSICAL CONTACT INFORMATION, ONLINE CONTACT INFORMATION, DEMOGRAPHIC DATA
INFO.
SHIPPING

Fig.8A.

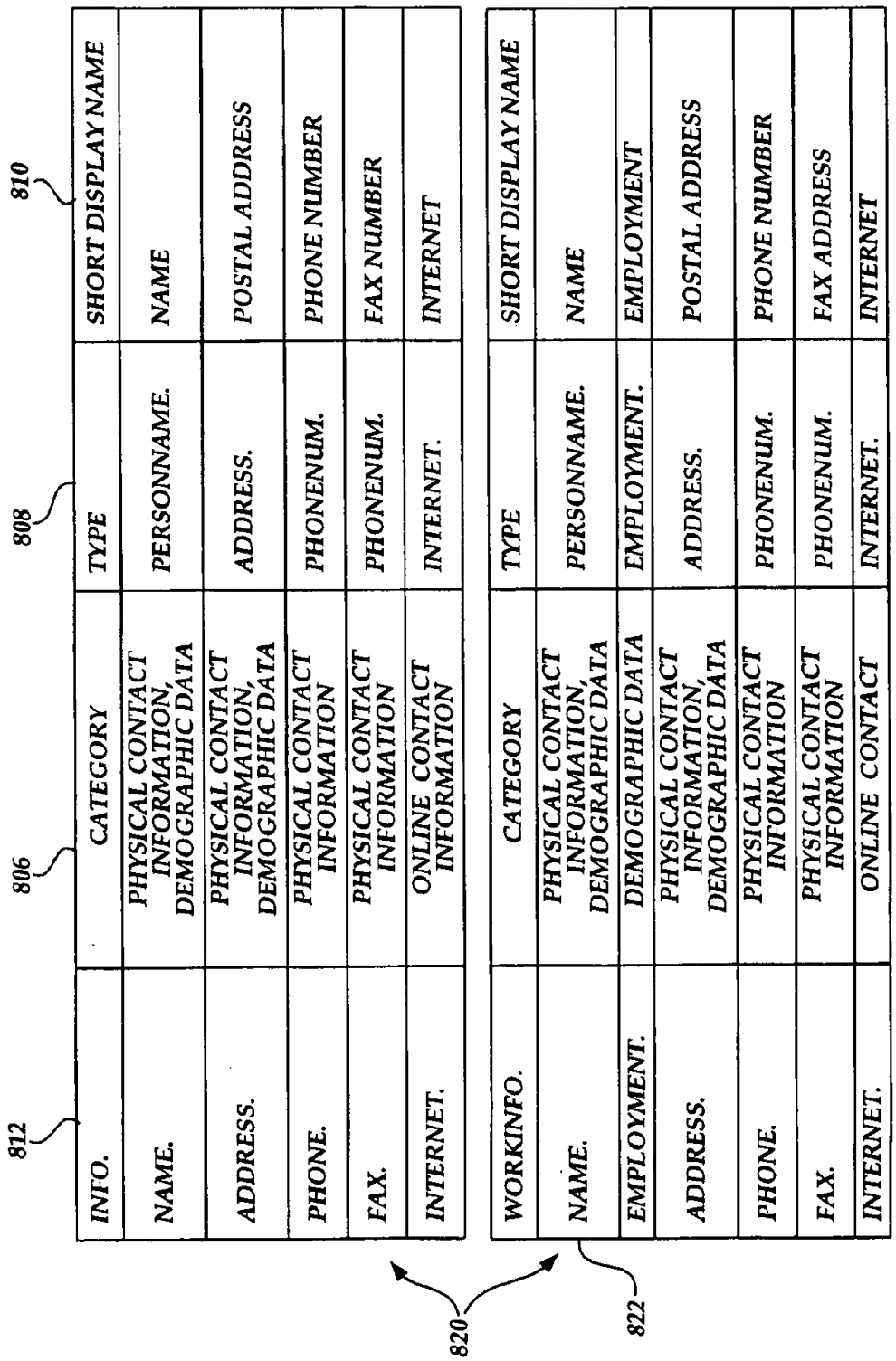

812
806
808
810
INFO.
CATEGORY
TYPE
SHORT DISPLAY NAME
NAME.
PHYSICAL CONTACT INFORMATION, DEMOGRAPHIC DATA
PERSONNAME.
NAME
ADDRESS.
PHYSICAL CONTACT INFORMATION, DEMOGRAPHIC DATA
ADDRESS.
POSTAL ADDRESS
PHONE.
PHYSICAL CONTACT INFORMATION
PHONENUM.
PHONE NUMBER
FAX.
PHYSICAL CONTACT INFORMATION
PHONENUM.
FAX NUMBER
INTERNET.
ONLINE CONTACT INFORMATION
INTERNET.
INTERNET
820
822
WORKINFO.
CATEGORY
TYPE
SHORT DISPLAY NAME
NAME.
PHYSICAL CONTACT INFORMATION, DEMOGRAPHIC DATA
PERSONNAME.
NAME
EMPLOYMENT.
DEMOGRAPHIC DATA
EMPLOYMENT.
EMPLOYMENT
ADDRESS.
PHYSICAL CONTACT INFORMATION, DEMOGRAPHIC DATA
ADDRESS.
POSTAL ADDRESS
PHONE.
PHYSICAL CONTACT INFORMATION
PHONENUM.
PHONE NUMBER
FAX.
PHYSICAL CONTACT INFORMATION
PHONENUM.
FAX ADDRESS
INTERNET.
ONLINE CONTACT
INTERNET.
INTERNET

Fig.8B.


INFORMATION
BILLINFO.
NAME.
ADDRESS.
PHONE.
FAX.
INTERNET.
CREDITCARD.
820
CATEGORY
PHYSICAL CONTACT INFORMATION, DEMOGRAPHIC DATA
PHYSICAL CONTACT INFORMATION, DEMOGRAPHIC DATA
PHYSICAL CONTACT INFORMATION
PHYSICAL CONTACT INFORMATION
ONLINE CONTACT INFORMATION
FINANCIAL DATA
TYPE
PERSONNAME.
ADDRESS.
PHONENUM.
PHONENUM.
INTERNET.
CREDITCARD.
SHORT DISPLAY NAME
NAME
POSTAL ADDRESS
PHONE NUMBER
FAX ADDRESS
INTERNET
FAVORITE CREDIT CARD
PERSONNAME
PREFIX
FIRST
MIDDLE
LAST
SUFFIX
824
CATEGORY
DEMOGRAPHIC DATA
PHYSICAL CONTACT INFORMATION
PHYSICAL CONTACT INFORMATION
PHYSICAL CONTACT INFORMATION
DEMOGRAPHIC DATA
TYPE
TEXT
TEXT
TEXT
TEXT
TEXT
SHORT DISPLAY NAME
PREFIX
FIRST NAME
MIDDLE NAME
LAST NAME
SUFFIX

Fig.8C.


ADDRESS
STREET1
STREET2
CITY
STATEPROV
POSTALCODE
COUNTRY
CATEGORY
PHYSICAL CONTACT INFORMATION
PHYSICAL CONTACT INFORMATION
PHYSICAL CONTACT INFORMATION
PHYSICAL CONTACT INFORMATION
DEMOGRAPHIC DATA
DEMOGRAPHIC DATA
TYPE
TEXT
TEXT
TEXT
TEXT
TEXT
TEXT
SHORT DISPLAY NAME
STREET1
STREET2
CITY
STATE/PROVINCE
POST/ZIP CODE
COUNTRY
824
PHONENUM.
AREACODE
NUMBER
EXTENSION
CATEGORY
PHYSICAL CONTACT INFORMATION
PHYSICAL CONTACT INFORMATION
PHYSICAL CONTACT INFORMATION
PHYSICAL CONTACT INFORMATION
TYPE
TEXT
TEXT
TEXT
TEXT
SHORT DISPLAY NAME
AREA CODE
NUMBER
EXTENSION

Fig.8D.


INTERNET.
EMAIL
HOMEPAGE
CATEGORY
ONLINE CONTACT INFORMATION
ONLINE CONTACT INFORMATION
TYPE
TEXT
TEXT
SHORT DISPLAY NAME
EMAIL
HOMEPAGE
EMPLOYMENT.
EMPLOYER
DEPARTMENT
JOB TITLE
CATEGORY
DEMOGRAPHIC DATA
DEMOGRAPHIC DATA
DEMOGRAPHIC DATA
TYPE
TEXT
TEXT
TEXT
SHORT DISPLAY NAME
EMPLOYER
DEPARTMENT
JOB TITLE
CREDITCARD.
TYPE
NUMBER
EXPMONTH
EXPYEAR
CATEGORY
FINANCIAL DATA
FINANCIAL DATA
FINANCIAL DATA
FINANCIAL DATA
TYPE
TEXT
TEXT
TEXT
TEXT
SHORT DISPLAY NAME
CARD TYPE
ACCOUNT NUMBER
EXPIRATION MONTH
EXPIRATION YEAR
824

Fig.8E.

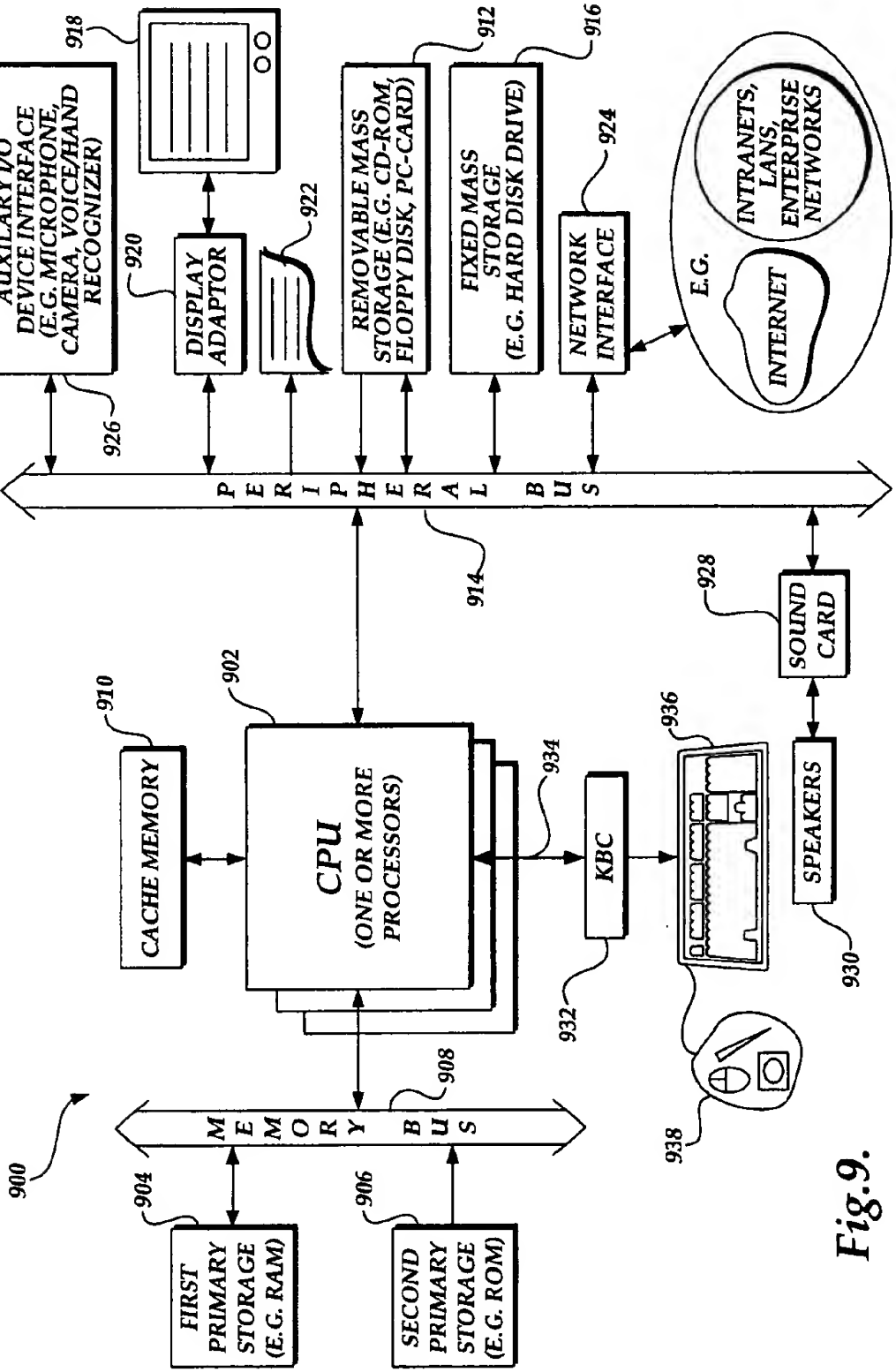

AUXILARY I/O DEVICE INTERFACE (E.G. MICROPHONE, CAMERA, VOICE/HAND RECOGNIZER)
DISPLAY ADAPTOR
REMOVABLE MASS STORAGE (E.G. CD-ROM, FLOPPY DISK, PC-CARD)
FIXED MASS STORAGE (E.G. HARD DISK DRIVE)
NETWORK INTERFACE
E.G.
INTRANETS, LANS, ENTERPRISE NETWORKS
INTERNET
PERIPHERAL BUS
SOUND CARD
SPEAKERS
CACHE MEMORY
CPU (ONE OR MORE PROCESSORS)
KBC
MEMORY BUS
FIRST PRIMARY STORAGE (E.G. RAM)
SECOND PRIMARY STORAGE (E.G. ROM)
900
902
904
906
908
910
912
914
916
918
920
922
924
926
928
930
932
934
936
938

Fig.9.

SERVER FOR ENABLING THE AUTOMATIC INSERTION OF DATA INTO ELECTRONIC FORMS ON A USER COMPUTER

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to computer software for filling out form documents over a computer network. More particularly, the present invention provides a method and system for automatically filling out fields in an electronic form document on a browser program using a remote server.

2. Discussion of Prior Art

Rapid growth and technological advances have changed the way most people currently use computers. During the early days of computers, a paradigm existed whereby there were more computer users than computers, and thus most computers had many assigned or dedicated users. As technology progressed, the personal computer ("PC") emerged, and it became commonplace for many computers to have only one user. Subsequent growth, particularly in the 1990s, has seen a culture or paradigm emerge whereby a computer user has access to more than one computer. As such, many individuals now have substantial access to multiple computers, for example at workplaces, schools, libraries, homes, and while traveling. This ratio of available computers per computer user should increase even further over time. It is therefore increasingly desirable to have computer-based programs and services that are accessible to a particular user from any computer, and not just those computers that have been programmed or adapted for that particular user.

One result of the recent explosion in computer growth is the amount of communication that now takes place between separate computers or computer systems. Many methods and systems exist for communications between computers or computer systems. This is reflected in many contexts, such as in the growth of the Internet. For purposes of the following discussion, several methods and systems will be described with reference to the Internet as a matter of convenience. It should be understood, however, that this is not intended to limit the scope of this discussion, and that many other applicable devices and protocols for computer communications exist, such as "Intranets", closed proxy networks, enterprise-wide networks, direct modem to modem connections, etc.

A browser program capable of running one or more windows may utilize a simple process for communicating information among computers over the Internet, as illustrated in FIG. **1**. Typically, an independent Internet user **106**, from a pool of random independent Internet users **101–106**, opens a browser window **131** in an Internet browser program, depicted by arrow **161**. User **106** then enters a request for an Internet Web page **144** (i.e. an HTML page) to be downloaded into browser window **131** belonging to user **106**. User **106**'s request is processed by the browser program, and a connection, depicted by arrow **162**, is made with the appropriate remote Internet resource **112**, typically an Internet Web server, selected from a pool of random remote **20** Internet resources **110–112**. Remote Internet resource **112** returns an HTML document **143**, depicted by arrow **163**. HTML document **143** contains substantially the entire content needed to display completed Web page **144**. Web page **144** is then displayed back to user **106** in browser window **131**, depicted by arrow **164**.

A model known in the art as the "ad server" model advances this simple browser program method for communicating information over the Internet. Many Internet Web pages are composite pages, requiring information in the form of images, text, and/or code to be pulled from several different remote Internet resources. Ad servers are generally used to integrate directed electronic material, such as banner advertisements, into such composite Internet Web pages. Thus, ad servers are one example of a remote Internet resource that separately contributes material to a composite Web page. A computer network communication process utilizing an ad server is also depicted in FIG. **1**.

Independent internet user **102** opens a browser window **130** in his or her Internet browser program, depicted by arrow **151**. User **102** then enters a request for an Internet Web page **142** to be downloaded into browser window **130**. This request is processed by the browser program, and a connection, depicted by arrow **152**, is made with the appropriate remote Internet resource **110**, typically an Internet Web server. Remote Internet resource **110** returns a core HTML document **140**, depicted by arrow **153**. Core document **140** contains some displayable content and an additional link to another image **141**, in this case a banner advertisement, stored at a separate remote Internet resource **120**, in this case an ad server. The browser program parses core document **140** to find and use this link to retrieve image **141**. The browser program then makes a connection, depicted by arrow **154**, with remote Internet resource **120** to retrieve image **141**. Remote Internet resource **120** returns image **141** to the browser program, depicted by arrow **155**. Image **141** is then merged with the displayable content of core document **140** to comprise completed Web page **142**. Web page **142** is then displayed back to user **102** in browser window **130**, depicted by arrow **156**. It should be appreciated that this process may be repeated many times for many separate portions of a particular Web page. In fact, many Web pages contain links to dozens of separate remote Internet resources, requiring this process to be repeated for each separate link.

Many remote Internet resources assign a specific user identifier containing state information, referred to as a session identifier or "cookie", to each particular user whenever a user connects to the resource, for example to retrieve an Internet Web page. This cookie is deposited into the user's browser program, which is instructed to show the cookie to the resource upon subsequent visits so that the resource can identify the user. The cookie conveys to the resource who the user is and what document or component thereof that the user wants. Use of these cookies is vital when components are assembled from various remote Internet resources into one integrated Web page, as a resource for a core HTML document may require several visits or communications from a Web browser while a page is assembled. Without such cookies, use of composite Web pages would be substantially hindered. Many resources assign temporary cookies for this purpose, which expire at the end of a session or when the browser program is closed. Other cookies, however, are assigned for longer durations for identification purposes beyond one Internet session. One such purpose identifies users, through long-term or persistent cookies, to specific user history and preferences, such that information, for example content specific banner advertisements related to such user history and preferences, may be directed at identified users in the future.

Proxy systems generally group many individual computers and computer users into a single network. This network is typically served by a single proxy server, which serves as a conduit for all communications among individual network users and between any individual network user and any

outside remote electronic resource or user. A proxy server may provide several useful functions, such as faster communication between network users, firewall protection for all network computers, and large and efficient storage. One example of efficient storage by a proxy server occurs when one network user requests a Web page from a remote resource that has recently been retrieved by another network user. Rather than retrieve the same Web page from the same remote resource again, the proxy server may simply deliver the Web page that has been stored on it from the previous request. It should be appreciated that while the use of proxy servers alters the path through which information and communication may travel, such use does not substantially alter the basic functions of the methods and systems described herein.

In general, many methods may be used to assist a user in filling out an electronic form document. One such method is referred to as the "wallet" method, which may be found, for example, at "eWallet" located at http://www.ewallet.com. This method requires a user to download a software application and install it onto his or her computer. The user is then required to input personal information items into the application, including the user's name, address, and credit card information, which is stored on the user's computer for future use. The user is then able to use this "eWallet" application and inputted personal information items to automatically fill out electronic form documents that are affiliated with the "eWallet" application. Whenever the user desires to fill out an affiliated electronic form document, the user opens the "eWallet" application and clicks and drags a virtual credit card from the virtual wallet onto the form document. The "eWallet" application then automatically inserts the inputted personal information items into the document. The click and drag step must be repeated for each page of the electronic form document. The user is then able to review and approve the form document before transmitting it as complete.

Another method for assisting a user in filling out an electronic form document is referred to as the "transactor" method, which may be found, for example, at "Transactor Networks" located at http://www.transactor.net. This method differs from the wallet method in that the user is not required to download or install any software onto his or her computer. Instead, personal information items are input and stored in a database on a remote server, which is then accessed every time an electronic form document is to be filled. This remote server containing the personal information items is accessed through a browser window separate from the browser window containing the electronic form document to be filled. This method thus requires communication between separate browser windows.

A method for filling out an electronic form document using this transactor method is illustrated in FIG. **2**. Independent Internet user **202**, from a pool of random independent Internet users **201–206**, opens a browser window **230** in his or her Internet browser program, depicted by arrow **251**. User **202** then enters a request for a Web page containing an electronic form document **240** to be downloaded into browser window **230**. This request is processed by the browser program, and a connection, depicted by arrow **252**, is made with the appropriate remote Internet resource **210**, typically an Internet Web server, selected from a pool of random remote Internet resources **210–212**. Remote Internet resource **210** returns an HTML Web page containing electronic form document **240**, depicted by arrow **253**. This process may include the retrieval of other portions of the Web page from separate remote electronic resources, as described above in the ad server model. User **202** also opens another separate browser window **231** to activate the "transactor" process, typically done through a bookmark. The browser program then makes a connection, depicted by arrow **255**, with transactor server **220** to retrieve the user's personal information items **241**. Transactor server **220** returns personal information items **241** to the browser program in browser window **231**, depicted by arrow **256**. It should be noted that the process represented by arrows **254** through **256** may be completed before or after the process represented by arrows **251** through **253**. Once both processes are complete, window **231** initiates communication with window **230** to begin the automated fill of electronic form document **240** in window **230**. The windows communicate as necessary until form **240** is filled, depicted by double arrow **257**. Filled electronic form document **242** is then displayed back to user **202** in browser window **230**, depicted by arrow **258**. User **202** is then able to review and approve filled electronic form document **242** before transmitting it as complete.

There are, however, several drawbacks to both the "wallet" and "transactor" methods. Both methods require a substantial initial time investment in requiring a user to input all of his or her personal information items. In addition, the wallet method requires the user to download and install software to his or her computer. This is problematic in that the user has no access to this method when he or she uses any computer that does not have the wallet program installed and the user's personal information items inputted. It is not only inconvenient to re-install and re-input items on every computer for which the user has access, but it is practically impossible for a computer being used by a user for the first time to have the user's personal information items. While the transactor method attempts to overcome this deficiency of the wallet method through server-side databases, it requires cross-communication between windows, which is known in the art to compromise user security. In addition, the requirement of retrieving information from a server-side database during window cross-communication significantly slows down the automated electronic form document fill process.

As such, what is desired is a method and system for remote server-based applications to quickly and automatically fill out electronic form documents, thereby relieving the user of the burden of manually inputting data into such form documents and without requiring the user to be on any specific computer and without compromising user security. In other words, what is desired is a method and system which enables a computer user to automatically fill out electronic form documents from any computer or client location in a network at the simple click of a mouse. Also desirable and inherent in such a method and system is the ability to automatically fill out electronic form documents without requiring the user to download or install any type of permanent or resident software on any computer.

SUMMARY OF THE INVENTION

According to the present invention, methods, apparatus, and computer program products are disclosed for constructing and transmitting an executable software module on a personal information server, referred to as a privacy bank server, to a remote computer. The software module is constructed such that once received by a browser displaying a form, it is executed and user data is automatically inserted into the form. In one aspect of the present invention, a method for constructing a shippable software module on a personal information server suitable for execution on a

remote computer for inserting data strings into an electronic form is described. A form mapping containing a set of associations between fields in the electronic form ("non-standard fields") and pre-named fields ("standard fields") on the personal information server is retrieved. Each mapping is associated with a registered electronic form. A raw data file containing data strings, each data string corresponding to a pre-named field is retrieved. Each raw data file is associated with a registered user. The form mapping is utilized to attach a data string to the field in the electronic form where the pre-named field and the field in the electronic form have been previously matched or mapped.

In one embodiment, an intended-practice condition associated with each non-standard field in the electronic form is compared to a use-preference condition associated with each pre-named or standard field. A data string is attached to the non-standard field in the electronic form when the intended-practice condition is less than or the same as the use-preference condition of the pre-named field. In another embodiment, the data string is not attached to the field in the electronic form when the intended-practice condition is greater than the use-preference condition of the pre-named field. In yet another embodiment, the form mapping is utilized to attach a data string to the field in the electronic form involving an examination of multiple negotiation objects for either an acceptance and decline message, where each negotiation object results from comparing data relating to the field in the electronic form to data relating to the pre-named field.

In another aspect of the invention, a server for enabling automatic insertion of user information into an electronic form having multiple fields on a remote computer capable of communicating with the server is described. The server contains a memory area storing a multiple raw data profiles where each raw data profile corresponds to a registered user of the privacy bank service. Another memory area stores multiple form mappings, each form mapping corresponding to a particular form registered with privacy bank service by a merchant or third-party vendor. A comparison module compares or "negotiates" user-preference data contained in the raw data profiles with practice-preference data contained in the form mappings. A software module constructor prepares and transmits a shippable program or software module that can be executed on a remote computer thereby inserting data strings into an electronic form on the remote computer.

In one embodiment, the raw data profile includes several standard field names, each standard field name having a corresponding data string and a use-preference data item determined by a registered user. Similarly, each form mapping includes multiple non-standard field names from the electronic form, where each non-standard field name is mapped to a standard field name and has a practice-preference data item. In yet another embodiment, a negotiation history module contains negotiation modules, each negotiation module containing an offer component and either an acceptance component or a decline component.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be better understood by reference to the following description taken in conjunction with the accompanying drawings in which:

FIG. **1** is a diagrammatical representation of an "ad server" model in accordance with the prior art.

FIG. **2** is a diagrammatical representation of a transactor model in accordance with the prior art.

FIG. **3A** is a diagrammatical representation of a system for automatically filling in electronic form documents in accordance with one embodiment of the present invention.

FIG. **3B** is a block diagram showing components of a server enabling the automatic insertion of data in to an electronic form on a remote computer in accordance with one embodiment of the present invention.

FIGS. **4A** and **4B** are flow diagrams of a process for automatically filling in an electronic form document in accordance with one embodiment of the present invention.

FIG. **5** is a flow diagram of a process for an initial user session using the service for automatic electronic form completion in accordance with one embodiment of the present invention.

FIG. **6** is a flow diagram of a process for constructing and transmitting a shippable code segment from a server to a user computer in accordance with one embodiment of the present invention.

FIG. **7** is a flow diagram of a process for mapping through which form names are mapped to a user's raw data values in accordance with one embodiment of the present invention.

FIGS. **8A**, **8B**, **8C**, **8D**, and **8E** are table diagrams showing the field names and format of registered user data in accordance with one embodiment of the present invention.

FIG. **9** is a block diagram of a typical computer system suitable for implementing an embodiment of the present invention.

DETAILED DESCRIPTION

Reference will now be made in detail to a preferred embodiment of the invention. An example of the preferred embodiment is illustrated in the accompanying drawings. While the invention will be described in conjunction with a preferred embodiment, it will be understood that it is not intended to limit the invention to one preferred embodiment. To the contrary, it is intended to cover alternatives, modifications, and equivalents as may be included within the spirit and scope of the invention as defined by the appended claims.

A method and system for automatically filling in electronic forms on a computer and not requiring a user to download or install any resident software on the computer is described in the various figures. As the presence of merchants and services increases on the Internet, electronic commerce or e-commerce will grow. More and more consumers will resort to the Internet, for example, to purchase goods and services. This will typically require the consumer/user to provide at least some data to the merchant typically through filling out an electronic form having various fields, most commonly names, addresses, credit card numbers, phone numbers, etc. For consumers purchasing goods from numerous merchant sites and possibly using different computers (e.g. using a computer at work, using another computer at home, and yet another one while travelling), manually filling in these forms repeatedly can become tedious and inefficient. The present invention seeks to alleviate the burden of filling in electronic forms, while informing the consumer/user of privacy precautions taken by a particular merchant site, and not require the user to download any resident software. Inherent in the latter feature is allowing the consumer to use the processes of the present invention from any computer connected to the network, the Internet in particular.

The present invention uses a remote server or "privacy bank," a novel electronic resource that responds to requests for data by preparing and transmitting a specialized document in the form of a JavaScript. This JavaScript is formed dynamically by the privacy bank upon receipt of the request

for data. The JavaScript created by the privacy bank is a "profile" or mapping between field names in a particular form the user needs to fill in at a particular merchant site (e.g. "www.fishermanstore.com") and standardized field names stored in the privacy bank server. Once the user's browser program is served this profile from privacy bank, most of the fields in the fishermanstore form are automatically filled in. In the described embodiment, the user becomes a member of the privacy bank resource by providing personal information, also referred to as the raw data, to privacy bank once. This raw data can be updated from time to time by the user if desired. In another embodiment, the user can enter privacy rules or requirements once when initially becoming a member. The user does not need to download any software from privacy bank or any other resource. In the described embodiment, the merchant (e.g. The Fisherman Store) becomes an affiliate member of the privacy bank network by providing a sample document of its form or forms. Privacy bank can then build a mapping between fields in the merchant's form and the standardized fields in its own database. These processes, components, and related data constructs are described in the various figures below.

FIG. **3**A is a block diagram of a system for automatically filling out electronic form documents in accordance with one embodiment of the present invention. A number of end-user computers are shown on the right side of the diagram. These computers can be client computers in a network with access to the Internet or be part of an intranet. In the described embodiment an end-user computer **302** is a stand-alone computer with access to the Internet and contains an Internet browser program, a browser window for which is shown at **304**. In the center of the diagram is a number of Web servers. A particular Web server **306** is a server for a merchant Web site, such as www.fishermanstore.com to which users or consumers can visit to purchase goods online over the Internet. On the left side of the diagram is a specialized electronic resource referred to in the described embodiment as a privacy bank server computer **308**, also connected to the Internet.

The process of automatic electronic form completion begins with a user downloading the form from a Web site such as fishermanstore site. A process of a user becoming a member and logging into the privacy bank server is described in greater detail in FIG. **5**. Returning to FIG. **3**A, a user/consumer on computer **302** ("user **302**") opens a browser window **304** in an Internet browser program such as Netscape Navigator or Internet Explorer, depicted by arrow **310**. User **302** then goes to www.fishermanstore.com shown by arrow **312** via the browser and decides to purchase goods. User **302** then downloads from the Web site contained on Web server **306** an electronic purchasing form that needs to be completed as depicted by arrow **314**. A purchasing form **316**, typically an HTML document, is returned and downloaded into and displayed in browser window **304**. At this juncture, user **302** would normally have to "manually" fill in each field in purchasing form **316**. Much of this information is typically standard: name, address, phone number, payment method, user email address, etc. In accordance with one embodiment of the present invention, user **302** can "click" on a privacy bank icon or button in form **316** and have the form automatically filled in.

As stated earlier, it is assumed in this discussion that www.fishermanstore.com is registered with and thus an affiliate member of the privacy bank service assessable from privacy bank server **308**. Being an affiliate member of the privacy bank service, purchasing form **316** contains a privacy bank icon or button **318**. By clicking on privacy bank icon **318**, user **302** essentially completes a process for automatically filling in form **316** by transparently transmitting a completed form to the privacy bank service on server **308**, depicted by an arrow **320**. The information needed for filling in the form is transmitted to user **302** once form **316** (an HTML document) is parsed, which occurs when form **316** is downloaded. This process is described in greater detail in FIGS. **4**A and **4**B. User **302** informs privacy bank server **308** of the identity of the user and of which Web site and which form on that Web site (if more than one) the user wishes to have filled in. This information is transmitted to privacy bank server **308**, unbeknownst to user **302**, when form **316** is downloaded. Techniques for accomplishing this are described below. Once privacy bank server **308** receives a request from registered user **302** (by virtue of an external link in form **316** executed when the form is parsed by user **302**), it begins preparing information needed to fill in form **316** on user computer **302**. A process of preparing the information sent back to user computer **302** and browser **304**, depicted by arrow **322**, is described in greater detail in FIGS. **6** and **7**. In the described embodiment, the information sent to user computer **302** is a JavaScript program **324** referred to as a "profile." Explained briefly, this profile contains a mapping of privacy bank standardized fields and fields in purchasing form **316** and "raw," generally personal, data associated with user **302**. The content of this profile and JavaScript program in general is described in greater detail in FIGS. **7**A and **7**B below. Once received by the browser program on user computer **302**, the filled out purchasing form **316** is displayed to user **302** as depicted by arrow **326**. This occurs when user **302** presses or clicks on privacy bank icon **318**. The information needed to complete form **316** is already resident in the browser program. At this juncture, user **302** can decide whether to proceed with submitting the form (typically after filling out a few more fields such as which items to purchase, quantity, etc.) or declining to submit the form, perhaps after reviewing the fishermanstore's Web site privacy safeguards or for any other reason.

FIG. **3**B is a block diagram showing components of a privacy bank server enabling the automatic filling in of electronic forms on a remote user computer. A privacy bank server, such as server **308** in FIG. **3**A, contains several functional and storage components needed for compiling the data needed for filing in a form, such as form **316**. Shown in FIG. **3**B are four major components of a privacy bank server in the described embodiment. These components and storage areas include a raw data profile storage area **328**, a form mapping storage area **330**, a negotiation history module **332**, and a shippable code constructor **334**. Raw data profile storage area **328** contains sets of data relating to registered users of the privacy bank service, one set or profile shown in area **336**. A registered user has a unique account number that can be used as an identifier and a password, shown in an area **338**. The standard field names set by the privacy bank service, discussed in greater detail in FIGS. **8**A, **8**B, **8**C, **8**D, and **8**E, are paired with a user entered data string (such as first name or home street address), followed by a use-preference condition. The use-preference condition is used in negotiation history module **332** and is discussed in greater detail in FIG. **7** below. This data is contained in an area **340**. Another profile for another registered user is shown in an area **342**. Each registered user has a similar raw data profile.

Form mapping area **330** includes multiple form mappings, an example of which is shown in an area **344**. Each electronic form that is registered with the privacy bank

service by an online merchant or seller (i.e. an affiliate member) has a form mapping. A privacy bank standard field name, as discussed below in FIGS. 8A, 8B, 8C, 8D, and 8E, and as mentioned above in area **340**, is matched or mapped with a "non-standard" field name from the electronic form registered with the service. For example, a non-standard field name for a person's last name could be "Last Name," "Surname" or simply "Last." Different forms use different variations of names for this field and for other fields. This would be mapped against the privacy bank "standard" field name, which in the described embodiment, is "Person-Name.Last." Also contained in area **346** is a practice-preference condition provided by the online merchant or seller when registering the form. As with the use-preference condition in area **340**, this condition is used by negotiation history module **350** and shippable code constructor **334**, and is discussed in greater detail below in FIG. **7**. Another mapping **348** having the same format for another registered form follows area **344**.

Negotiation history module **332** is used to determine which fields in the electronic form will be automatically filled in by the privacy bank server. A process associated with negotiation history module **332** is described in greater detail in FIG. **7**. Module **332** includes multiple negotiation objects, an example of which is shown in an area **350**. In the described embodiment, each negotiation object corresponds to one "non-standard" field in the form. Described briefly, negotiation object **350** contains information as to whether the field in the form should be filled in based on privacy and use preferences set by the user (as conveyed in use-preference condition in area **340**) and compared to intended practices (as conveyed by practice-preference condition in area **346**). This comparison is performed in the negotiation history module, which includes a negotiator or comparator for comparing these conditions.

Specific conditions in the described embodiment are described below. If it is determined that the non-standard field in the form will be filled in, a data string, shown in area **340**, will be included in negotiation object **350**. Shippable code constructor **334** accesses component **332** and storage areas **328** and **330**, to derive a software module to be transmitted to a user computer. In the described embodiment, the software module is a JavaScript program which is transmitted to and executed by a browser in the user computer, thereby inserting the data strings into the form fields.

FIGS. **4A** and **4B** are flow diagrams of a process for automatically filling in electronic forms in a computer network in accordance with one embodiment of the present invention. The process described below can be performed in a configuration of servers and computers as described in FIG. **3A** above. At a step **402** an online user/consumer desiring to purchase certain goods on the Internet downloads an electronic form for making a purchase into the user's browser program. At step **404** the browser parses the electronic form content, typically an HTML document, to identify all external links. As is commonplace for Web pages, the HTML contains links to other external Web sites from which content or other types of data is retrieved. In many instances, a Web page is a composite of different components from various sites embedded in a core HTML document. An example is an external link to an ad server to retrieve a banner ad component of a core HTML document. In this case, the electronic form can be seen as a core HTML document. This parsing is done automatically by the browser and is a well known feature.

At step **406** the browser identifies an external link to the privacy bank server. In the described embodiment, this link will necessarily be present since the Web site is an affiliated site of the privacy bank service network of registered sites. A description of what "registered" implies in this context is described in greater detail below. At step **408**, the browser executes the external link to connect the browser to the privacy bank server. The external link is referred to in the described embodiment as a shippable code link to the privacy bank server. The shippable code in this context is a JavaScript program that is transmitted from the privacy bank server to the user computer and browser. This shippable code enables the electronic form to be filled in automatically in a process that is described in greater detail below. At step **410**, once the privacy bank server has been contacted via the shippable code link in the electronic form, the privacy bank server determines whether the user computer or browser has a valid state identifier, referred to as a "cookie", previously assigned to it by the privacy bank server. A cookie is an identifier assigned by a Web site, whether a Web server or a server such as the privacy bank server, to a user/visitor when the user visits the Web site for the first time in a given session (the time from which a user logs onto the Web and the time he or she exits the Web by exiting the browser). The cookie, assigned by a Web site, belongs to a particular user. In the described embodiment, the user keeps this cookie during its session (a temporary cookie) and if the user goes back to that Web site during that session, it shows the Web site that cookie from which the Web site can identify the user and retrieve characteristics of that user from its data repository. As is known in the art of Internet application programming, cookies can also be permanent in that they subsist with a user after the user has logged off the browser and can be used again in a new session. The concept and implementation of cookies themselves are well known in the field of Internet and, more generally, computer network programming.

If the privacy bank server determines that the user computer or browser does not have a valid cookie, it implies that the user has not yet logged into the privacy bank service. If so, control goes to step **412** where the privacy bank server retrieves a login sequence code. This code will trigger a login sequence and enable the user to log in to or register with the privacy bank service at a later step in the process, as described in greater detail below. At step **414**, the privacy bank server forms a completed package of shippable code containing the retrieved login sequence code, such that the login sequence will be triggered at a later step in the process. At step **422**, the browser retrieves this completed package of shippable code from the privacy bank server. The shippable code is then stored in the browser residing on the user's computer, and is executable upon a user trigger, which is described in greater detail below.

If the privacy bank server determines that the user/browser making contact by downloading the electronic form and executing the external link has a valid cookie, control goes to step **416** where the privacy bank server gets and reads the user's cookie. In this context, having a valid cookie indicates that the user has already gone through the login sequence recently, for example during the existing Internet session, and thus it is not necessary for the user to go through the login sequence again. By reading the user's cookie, the privacy bank server can determine who the user is and thus can retrieve the user's raw data stored by privacy bank. The contents and format of this raw personal data is described in greater detail in FIGS. **8A**, **8B**, **8C**, **8D**, and **8E** below. At step **418**, the privacy bank server retrieves the user data for the user identified by the valid cookie. The privacy bank server couples this user data and an identifier, such as a URL

(uniform resource locator), to determine how the electronic form document should be filled. At step **420**, the privacy bank server forms a completed package of shippable code (item **324** in FIG. **3**A) containing the user data that will be used to fill out the form document. In the described embodiment, this shippable code, referred to as a profile, is in the form of a JavaScript program. This shippable code is formed from information in the privacy bank memory that will enable the form document to be filled out automatically at a step later in the process. At step **422** the browser receives the shippable code, or profile, from the privacy bank server, and now has access to it on the user computer, if desired by the user. This profile is stored in the browser residing on the user's computer, and is executable upon a user trigger.

Assuming the user wants to have the electronic form automatically filled out using privacy bank, he or she executes a user trigger. In the described embodiment, this trigger occurs when the user clicks on an "autofill" button contained in the form and associated with privacy bank at step **424**. By clicking on the autofill button, the user allows the browser to execute the shippable code or profile stored thereon. At step **426**, the shippable code determines whether it contains user data which would permit it to fill out the form document. If user data is contained within the shippable code residing on the browser, control goes to step **434** where the browser utilizes the shippable code and user data to fill out the electronic form document. Of course, user data being present in the shippable code is dependent upon the browser having a pre-existing valid cookie when the form document was initially retrieved, as described above.

If, however, user data is not contained within the shippable co residng on the browser, control goes to step **428** where the login sequence is initiated in order to identify the presently unknown user. The shippable code utilized by the browser in this step contains the login sequence code, which is a result of the browser not having a pre-existing valid cookie when the form document was initially retrieved, as described above. The login process of step **428** is described in greater detail in portions of FIG. **5**. Once the user completes the login sequence at step **428**, the privacy bank server assigns a cookie to the user/browser thereby enabling it to recognize the user and messages from the user's browser in subsequent transactions. At step **430**, the privacy bank server retrieves the user data for the identified user, couples this user data and an identifier, such as a URL (uniform resource locator), to determine how the electronic form document should be filled, and forms a completed package of shippable code containing the user data that will be used to fill out the form document. This step is substantially similar to steps **418** and **420**, as described above. At step **432**, the browser receives the shippable code, or profile, from the privacy bank server, and now has access to it on the user computer. This shippable code now contains user data that allows the form document to be filled out automatically. At this stage, control proceeds to step **434**, where the browser utilizes the shippable code and user data to fill out the electronic form document. Further input from the user, such as re-clicking on the "autofill" button, is not required. In other words, once the user properly completes the login sequence, the form is then filled out automatically, and it is not necessary for the user to click on the "autofill" button again.

At step **436**, the user visually examines the filled out form and the privacy features offered by the Web site and decides whether the form is acceptable. If the user finds that the form needs further adjustment, the user adjusts the document at step **438**. This may be done manually, or through any supplemental automated process, such as a client-based macro. This can involve filling in fields that could not be filled in by the profile sent by the privacy bank server (in other words, fields that could not be filled out from the raw data). Such fields can include, for example, which items being purchased and the quantity of items. It can also include updated personal information such as a new address or credit card number. In this case, the user simply types over the information already in the fields. Control then returns to step **436**, which is satisfied presumably after going through step **438** once. At step **440** the browser submits the filled out electronic form eventually sending it to the merchant's Web server once the user clicks on a Submit form button in the browser window. In the described embodiment, the filled out form is first sent to the privacy bank server unbeknownst to the user or at least transparent to the user. The completed form is received and examined by the privacy bank server which updates its raw data repository to reflect any changes the user may have made to his or her personal information. The privacy bank server then posts a message back to the user computer (according to HTTP protocol the server must send a message back to the user computer when it receives an HTML document from it). In the described embodiment, the message it sends back or posts to the user's browser is similar to a "Click Here To Continue" type screen to the user. Hidden behind this message is the completed form that was sent to the privacy bank server. Presumably, the user will click to continue and by doing so transmits the hidden completed form to the merchant's Web server. In other preferred embodiments, the completed form is sent to both the privacy bank server and the merchant's Web server at the same time. In yet another preferred embodiment, the completed form is posted automatically from the privacy bank server directly to the merchant's site without any additional input from the user. At this stage the automatic form filling process is complete.

FIG. **5** is a flow diagram of a process for entering new users into the privacy bank service or logging in existing members thereby allowing them to use the service in accordance with one embodiment of the present invention. Portions of FIG. **5** show in greater detail step **428** of FIG. **4**B where the login sequence is initiated to identify the user. Once the privacy bank server determines that the user does not have a valid cookie, the server inserts a login process for the user into the shippable code, whereby an account may be created if the user does not already have one. This is done because it is assumed that if the user has not yet been assigned a cookie from privacy bank for the user's current session, he or she has not yet logged onto the privacy bank service or is possibly not a registered member. The first event that occurs in the login sequence is the privacy bank server displaying a login screen in the user's browser window, as depicted in step **502**. At step **504** the user decides whether or not he or she is a privacy bank member, and proceeds to utilize the appropriate section of the login screen. One section of the login screen requires the user to input a user identification and password for an existing account, while another section permits the user to select an icon that routes the session to an account creation screen.

For account creation, in the described embodiment, the user selects a "New Account" icon on the login screen, as depicted in step **506**. At step **508**, a privacy bank account creation screen is then displayed. At step **510**, the user enters his or her user identification, a password, name, other information, and high-level privacy preferences into the account creation screen. Essentially the user is configuring the account and personal information items that will be

stored for him or her in the privacy bank repository. The information inputted into this newly created account is then posted back to the privacy bank server, as depicted in step **512**. At step **514**, the privacy bank server accepts the new account information and establishes a location for the new user in the privacy bank repository. The newly inputted information is then recorded in this repository location. The privacy bank server then sets a cookie for the current user session, as depicted in step **522**, and the process ends.

For existing user login, in the described embodiment, the user enters his or her existing user identification and password, as depicted in step **516**. At step **518**, this login information is posted back to the privacy bank server, which then proceeds to evaluate the information. At step **520**, the privacy bank server determines whether or not the posted information is correct, that is, whether or not it corresponds to a valid user identification with the proper password. If the posted information does not correspond to a valid user identification and password, then the attempted login fails, and the process reverts to step **502**, where a new login screen is displayed. If the posted information is correct, then the login is successful, and the process proceeds to step **522**. The privacy bank server then sets a cookie for the current user session, as depicted in step **522**, and the process ends.

FIG. **6** is a flow diagram of a process for deriving the parts needed in constructing a shippable code segment or profile to be posted to the user in accordance with one embodiment of the present invention. It describes a process leading up to step **416** of FIG. **4A** in which the browser retrieves the shippable code posted from privacy bank in greater detail. Recall that the user's browser parses the electronic form to identify and then execute any links to external resources to obtain external components to the core HTML document. In the case of the privacy bank server (assuming the merchant Web site from which the electronic form is being downloaded is an affiliate member of privacy bank), the server receives and begins to perform operations pursuant to the link from the user. At step **602** the user retrieves two items: the user cookie and the identifier of the electronic form the user presumably wants to fill out. The user cookie (assigned to the user by the privacy bank server from when the user logged onto the service) informs the privacy bank server of the identity of the user. The identifier of the electronic form contains an identifier of the merchant's Web site, and the specific form on the site that has been downloaded by the user, also in the form of a URL in the described embodiment. In many instances there may only be one form on the Web site.

At step **604** the privacy ban server uses the user cookie to retrieve the user's raw data from the privacy bank memory. The configuration and content of the raw data is described in greater detail in FIGS. **8A**, **8B**, **8C**, **8D**, and **8E** below. The raw data is a set of data items very likely needed to fill out common electronic purchasing forms. As is described in greater detail below, each raw data item corresponds to a particular privacy bank standard label or name. At step **606** the privacy bank server uses the URL or other identifier for the specific form to be filled out to retrieve a mapping of each field name in the electronic forms (i.e. the legacy names) to privacy bank standardized names. This mapping or field name matching is performed when a merchant becomes an affiliate member of the privacy bank service. At that juncture the merchant submits one or more forms to privacy bank which then examines each field name in the forms and matches it with a privacy bank field name. In the described embodiment, if there is a legacy name that does not have a matching privacy bank field name, the privacy bank user raw data configuration can be updated if it is believed that the particular legacy field name may begin appearing on other forms. Otherwise, it is left for the user to manually fill in as described in step **424** of FIG. **4A**.

At step **608** the server merges the retrieved name map and the user's raw data through a join type operation. The merger between two tuples: the legacy name/privacy bank name tuple and the privacy bank name/raw data value tuple is described in greater detail below. The outcome of this merger is a series of tuples matching a legacy name with a raw data value associated with the user. In other preferred embodiments this merger can be performed using other data constructs and operations. However, the outcome is a pairing of a legacy field name and a raw data value. At step **610** the series of tuples from the merger is converted to a shippable code. In the described embodiment the shippable code is in the form of a JavaScript program which is posted to the browser on the user computer. Normally, browser programs have a JavaScript component that is manipulable by JavaScript commands. These JavaScript commands in the shippable code are used to fill in the electronic form on the browser, a technique well known in the field of Internet and Java programming. It useful to note here that the form is actually filled in at the user computer via the browser using the JavaScript commands in the shippable code. The form is not filled out on the privacy bank server; the information for filling out the form is constructed there but is then posted to the user computer. At this stage the process of deriving the shippable code on the privacy bank server is complete.

FIG. **7** is a flow diagram of a process for mapping through which form names are mapped to a user's raw data values in accordance with one embodiment of the present invention. As described above, at step **702** the privacy bank server uses the form URL or other identifier to retrieve a mapping for the form that maps a legacy name with a privacy bank standardized name. The mapping also contains one or more practices for each field name, described in greater detail below. This mapping is created when a merchant or service provider decides to become an affiliate member of the privacy bank service. During the registration process the merchant tells privacy bank which forms it wants to register and the field names in those forms, which are then paired with privacy bank standardized names. At step **704** the user cookie is used to obtain the user's raw data, which includes actual data values and preferences associated with each data value. The practices mentioned above associated with legacy names and the preferences associated with user raw data values are stated in terms of conditions. In the described embodiment, there are various conditions, such as marketing (targeted), system administration, personalization, research and development, and completion of activity (i.e. ordering). Other embodiments can more or fewer conditions.

When a merchant registers with the privacy bank service, it must state what conditions for which it will use each legacy field. For example, for the field "Last Name" it may state that its practice will be to use this field for "personalization" and "completion of activity," and nothing else. For a "Method of Payment" field, it may state that its practice will be to use this field for "completion of activity," "research and development," and "administration" only. In this manner, the merchant will build a list of (legacy field name, practice) pairs stored in the privacy bank server. Similarly, when a user becomes a member of the privacy bank service, he or she is required to provide the raw data values (specific fields for the raw data are described below) and corresponding preferences, also stated in terms of conditions. They are called preferences because from the user's

point of view they indicate a privacy or use threshold. For example, a user can require that her last name can only be used for "personalization," " completion of activity," and "system administration," thereby excluding its use for targeted "marketing" for example. In the described embodiment, if a user does not specify a preference condition, the data item is not to be released, such as a social security number or a mother's maiden name.

At step **706** the privacy bank server retrieves a single (legacy name, practice) pair and its corresponding standardized privacy bank name from the form mapping retrieved at step **702**. In the described embodiment, this pair can be the first form field in the merchant's online form. At step **708** the server retrieves a corresponding (standardized privacy bank name, preference) pair from the user's raw data "file." The privacy bank name in this pair should match the privacy bank name in the pair retrieved at step **706**:

[(legacy name, practice), PB Name 1]: [PB Name 1, preference]

At step **710** the privacy bank server compares the merchant's practice conditions on the left with the user's preference conditions on the right. For example, for the last name field, the merchant has specified that its practice is to use this data item for the conditions "personalization" and "completion of activity." The user has specified she will only allow her last name to be used for the conditions "personalization," "completion of activity," and "system administration." The merchant's conditions and the user's conditions are compared. At step **712** the privacy bank server determines whether the merchant's form field should be filled in taking into account the user's privacy threshold for that field. In the described embodiment this is done by determining whether the user's preference conditions is a superset of the merchant's practice conditions. That is, does the merchant intend to use the user's last name for anything other than what the user has specified. In the last name example, the user's preferences is a superset of the merchant's practices: "personalization," "completion of activity," and "system administration" includes at least all of "personalization" and "completion of activity."

If the user's preferences are not a superset of the merchant's practices, control goes to step **714** where a message is displayed to the user indicating that the field will not be automatically filled in because the merchant may use that information in ways the user has not authorized. In the described embodiment, if one field in the form meets this condition, none of the fields in the form are filled in and the process is complete. In other preferred embodiments, the user has the option to fill in the field manually and have the privacy bank service proceed with the remaining fields.

If the user's preferences are a superset of the merchant's practices, control goes to step **716** where the field is filled in with the raw data value. Steps **710** and **712** can be seen as a two-step negotiation process. The merchant form, seen as an "information buyer," makes a proposal to the user, the "information seller." The proposal is essentially in the form of what data item the information buyer wants and what he intends to use it for. This is the first step in the negotiation process where the privacy bank server acts as a negotiator. The user gets the proposal and checks whether the merchant's conditions exceed what the user's preferences. In other words, the user checks whether the merchant intends to use the data item for purposes not specifically allowed by the user. If the merchant's practice conditions are acceptable (by performing the superset test in step **712**), the user sends an acceptance to the merchant, at which point the raw data value is retrieved and associated with the legacy field. If the merchant's conditions are not acceptable, the user essentially sends a "not accepted" message to the merchant through the negotiator without a raw data value. This completes the second step in the negotiation process. Each two-step negotiation is viewed as an object which is later used to construct a JavaScript program (the shippable code) using standard Java programming techniques.

At step **718** the server checks whether there are any other (legacy name, practice) pairs in the merchant's form. If there are more legacy fields, control returns to step **706** and the process repeats. If there are no more fields to be filled in, the process is complete. At this stage there is a series of objects or a history of negotiations that has been derived from the mapping process. These series of objects are then used to construct a JavaScript program. In the described embodiment, all the objects will have an acceptance from the user and thus a raw data value attached which is included in the JavaScript profile sent over to the browser/user. In other preferred embodiments, some of the objects can have a "not accepted" or declined message indicating that a particular field in the form will not be filled in and thus not have a raw data value. As mentioned above, in the described embodiment if one of the fields in the form cannot be filled in because the merchant's practices exceed the user's preferences, the entire form is not filled in. The shippable code or profile sent to the user represents a series of (legacy field name, raw data value) pairs. without any reference to preferences or practices, all the negotiations for the data values having been performed on the privacy bank server.

FIG. **8** is a high-level table diagram showing how fields containing the raw data and preferences for a user are organized on the privacy bank server in accordance with one embodiment of the present invention. A top-level User table **802** has four columns: User **804**, Category **806**, Type **808**, and Short display name **810**. User Table **802** has four areas of data under column User **804** represented by four rows: Home **812**, Work **814**, Billing **816**, and Shipping **818**. In the described embodiment, the user is presented with these four areas of data when registering with the privacy bank service and enters information by going through each of these data areas. Skipping Category **806** for the moment, column Type **808** takes the raw data tree down one level from the top level represented by table **802**. For example, the Type for data area Home **812** is Info. This performs as a pointer or link to an Info table **820**. The first column **822** of table **820** is labeled Info but the other three columns are the same as shown in table **802**; that is, Category **806**, Type **808**, and Short display name **810**.

At table **820**, the user begins entering data that will be used for her home information and for Shipping since data area **818** for Shipping in table **802** also has an Info in its Type column **808**. A Name row **822** has in its Type column **808** a reference to yet another table PersonName, shown as table **824**. Similar to table **802** and **820**, PersonName table **824** has a first column labeled PersonName and the same three columns as the other tables. All five data areas in PersonNamne table **824**: Prefix, First, Middle, Last, and Suffix have as a Type a primitive type referred to as Text in the described embodiment. Text represents a data string that is the actual data item stored in the privacy bank server. By examining the Type column **808** of each of the data areas, a user enters all the raw personal data. An actual data item is entered at each Type box containing Text, indicating a primitive type, or a leaf node when viewed as a tree structure. If the Type column does not contain "Text," another table exists that refines the data area further.

To follow another example, under the data area Billing **816** shown in table **802**, its Type **808** indicates BillInfo and

not Text. A table BillInfo has six further data areas, none of which have a Text Type, so no actual data values can be found at this level. Taking the CreditCard data area as an example, its Type indicates "CreditCard." Table CreditCard, shown in FIG. 8C, has four data areas: Type, Number, ExpMonth, and ExpYear, all of which are of Type Text, which contain actual data values.

Short display name column **810** contains a string that is displayed to the user through a user registration graphical user interface of the described embodiment. The user follows the data tree via a user interface using the Short display name strings as field names or guides to entering the data. The data areas that have primitive Types, which in the described embodiment is Text, are the privacy bank field names that are mapped with the legacy field names in the electronic forms registered with the service. In the described embodiment, the privacy bank names include (in abbreviated form):

PersonName.Prefix	Address.Street1	PhoneNum.AreaCode
PersonName.First	Address.Street2	PhoneNum.Number
PersonName.Last	Address.City	PhoneNum.Extension
PersonName.Suffix	Address.StateProv	
	Address.PostalCode	
	Address.Country	
Internet.Email	Employment.Employer	
Internet.HomePage	Employment.Department	
CreditCard.Type	Employment.JobTitle	
CreditCard.Number		
CreditCard.ExpMonth		
CreditCard.ExpYear		

Category column **806** is related to privacy settings set by the user and are tied to the preferences set by a user and defined in terms of the conditions as described above, specifically in FIG. 7. The conditions or use thresholds in the described embodiment are marketing (targeted), system administration, personalization, research and development, and completion of activity (i.e. ordering). The Categories available in the described embodiment and as shown in the tables of FIGS. 8A, 8B, 8C, 8D, and 8E, are Physical Contact Information, Online Contact Information, Demographic Data, and Financial Data. The relationship between the Categories and the conditions of the described embodiment can be described as a table five-row, four-column table (a 20 cell table) where each condition is one row in the table and each Category is one column in the table.

The present invention employs various computer-implemented operations involving data stored in computer systems. These operations include, but are not limited to, those requiring physical manipulation of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. The operations described herein that form part of the invention are useful machine operations. The manipulations performed are often referred to in terms, such as, producing, identifying, running, determining, comparing, executing, downloading, or detecting. It is sometimes convenient, principally for reasons of common usage, to refer to these electrical or magnetic signals as bits, values, elements, variables, characters, data, or the like. It should remembered, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

The present invention also relates to a device, system or apparatus for performing the aforementioned operations. The system may be specially constructed for the required purposes, or it may be a general purpose computer selectively activated or configured by a computer program stored in the computer. The processes presented above are not inherently related to any particular computer or other computing apparatus. In particular, various general purpose computers may be used with programs written in accordance with the teachings herein, or, alternatively, it may be more convenient to construct a more specialized computer system to perform the required operations.

FIG. **9** is a block diagram of a general purpose computer system **900** suitable for carrying out the processing in accordance with one embodiment of the present invention. FIG. **9** illustrates one embodiment of a general purpose computer system such as user computer **302** of FIG. **3**A and also describes many components found in privacy bank server **308**. Other computer system architectures and configurations can be used for carrying out the processing of the present invention. Computer system **900**, made up of various subsystems described below, includes at least one microprocessor subsystem (also referred to as a central processing unit, or CPU) **902**. That is, CPU **902** can be implemented by a single-chip processor or by multiple processors. CPU **902** is a general purpose digital processor which controls the operation of the computer system **900**. Using instructions retrieved from memory, the CPU **902** controls the reception and manipulation of input data, and the output and display of data on output devices.

CPU **902** is coupled bi-directionally with a first primary storage **904**, typically a random access memory (RAM), and uni-directionally with a second primary storage area **906**, typically a read-only memory (ROM), via a memory bus **908**. As is well known in the art, primary storage **904** can be used as a general storage area and as scratch-pad memory, and can also be used to store input data and processed data. It can also store programming instructions and data, in the form of data objects or JavaScript programs, for example, in addition to other data and instructions for processes operating on CPU **902**, and is used typically used for fast transfer of data and instructions in a bi-directional manner over the memory bus **908**. Also as well known in the art, primary storage **906** typically includes basic operating instructions, program code, data and objects used by the CPU **902** to perform its functions. Primary storage devices **904** and **906** may include any suitable computer-readable storage media, described below, depending on whether, for example, data access needs to be bi-directional or uni-directional. CPU **902** can also directly and very rapidly retrieve and store frequently needed data in a cache memory **910**.

A removable mass storage device **912** provides additional data storage capacity for the computer system **900**, and is coupled either bi-directionally or uni-directionally to CPU **902** via a peripheral bus **914**. For example, a specific removable mass storage device commonly known as a CD-ROM typically passes data uni-directionally to the CPU **902**, whereas a floppy disk can pass data bi-directionally to the CPU **902**. Storage **912** may also include computer-readable media such as magnetic tape, flash memory, signals embodied on a carrier wave, PC-CARDS, portable mass storage devices, holographic storage devices, and other storage devices. A fixed mass storage **916** also provides additional data storage capacity and is coupled bi-directionally to CPU **902** via peripheral bus **914**. The most common example of mass storage **916** is a hard disk drive. Generally, access to these media is slower than access to primary storages **904** and **906**.

Mass storage **912** and **916** generally store additional programming instructions, data, and the like that typically

are not in active use by the CPU **902**. It will be appreciated that the information retained within mass storage **912** and **916** may be incorporated, if needed, in standard fashion as part of primary storage **904** (e.g. RAM) as virtual memory.

In addition to providing CPU **902** access to storage subsystems, the peripheral bus **914** is used to provide access other subsystems and devices as well. In the described embodiment, these include a display monitor **918** and adapter **920**, a printer device **922**, a network interface **924**, an auxiliary input/output device interface **926**, a sound card **928** and speakers **930**, and other subsystems as needed.

The network interface **924** allows CPU **902** to be coupled to another computer, computer network, or telecommunications network using a network connection as shown. Through the network interface **924**, it is contemplated that the CPU **902** might receive information, e.g., data objects or program instructions, from another network, or might output information to another network in the course of performing the above-described method steps. Information, often represented as a sequence of instructions to be executed on a CPU, may be received from and outputted to another network, for example, in the form of a computer data signal embodied in a carrier wave. An interface card or similar device and appropriate software implemented by CPU **902** can be used to connect the computer system **900** to an external network and transfer data according to standard protocols. That is, method embodiments of the present invention may execute solely upon CPU **902**, or may be performed across a network such as the Internet, intranet networks, or local area networks, in conjunction with a remote CPU that shares a portion of the processing. Additional mass storage devices (not shown) may also be connected to CPU **902** through network interface **924**.

Auxiliary I/O device interface **926** represents general and customized interfaces that allow the CPU **902** to send and, more typically, receive data from other devices such as microphones, touch-sensitive displays, transducer card readers, tape readers, voice or handwriting recognizers, biometrics readers, cameras, portable mass storage devices, and other computers.

Also coupled to the CPU **902** is a keyboard controller **932** via a local us **934** for receiving input from a keyboard **936** or a pointer device **938**, and ending decoded symbols from the keyboard **936** or pointer device **938** to the CPU **902**. The pointer device may be a mouse, stylus, track ball, or tablet, and is useful for interacting with a graphical user interface.

In addition, embodiments of the present invention further relate to computer storage products with a computer readable medium that contain program code for performing various computer-implemented operations. The computer-readable medium is any data storage device that can store data which can thereafter be read by a computer system. The media and program code may be those specially designed and constructed for the purposes of the present invention, or they may be of the kind well known to those of ordinary skill in the computer software arts. Examples of computer-readable media include, but are not limited to, all the media mentioned above: magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROM disks; magneto-optical media such as floptical disks; and specially configured hardware devices such as application-specific integrated circuits (ASICs), programmable logic devices (PLDs), and ROM and RAM devices. The computer-readable medium can also be distributed as a data signal embodied in a carrier wave over a network of coupled computer systems so that the computer-readable code is stored and executed in a distributed fashion. Examples of program code include both machine code, as produced, for example, by a compiler, or files containing higher level code that may be executed using an interpreter.

It will be appreciated by those skilled in the art that the above described hardware and software elements are of standard design and construction. Other computer systems suitable for use with the invention may include additional or fewer subsystems. In addition, memory bus **908**, peripheral bus **914**, and local bus **934** are illustrative of any interconnection scheme serving to link the subsystems. For example, a local bus could be used to connect the CPU to fixed mass storage **916** and display adapter **120**. The computer system shown in FIG. **9** is but an example of a computer system suitable for use with the invention. Other computer architectures having different configurations of subsystems may also be utilized.

Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced within the scope of the appended claims. Furthermore, it should be noted that there are alternative ways of implementing both the process and apparatus of the present invention. For example, the raw data can contain more or few fields than those described as needed, and there can be additional privacy or use threshold conditions than the five described. In another example, the filled out electronic form can be sent automatically to the merchant's Web server after the privacy bank server updates its raw data without additional input from the user. In another example, the raw data and legacy fields can be bundled and coded in a software module other than as a JavaScript module. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims.

What is claimed is:

1. A method for constructing a shippable software module on a personal information server suitable for execution on a remote computer for inserting data strings into an electronic form, the method comprising:

receiving a request from a remote computer for a shippable software module suitable for execution on the remote computer for inserting data strings into the electronic form;

retrieving a form mapping containing a plurality of associations between fields in the electronic form and pre-named fields on the personal information server, the mapping being associated with the electronic form;

retrieving a raw data file containing data strings from a plurality of raw data files, each data string corresponding to a pre-named field and the raw data file being associated with a particular user;

dynamically constructing a shippable software module suitable for execution on the remote computer for inserting data strings into an electronic form utilizing the form mapping and the raw data file and comparing an intended-practice condition associated with each field in the electronic form as determined by a form-originating server, with a use-preference condition associated with each pre-named field as determined by the particular user; and

transmitting the shippable software module to the remote computer.

2. The method as recited in claim **1** further comprising attaching a data string from the raw data file corresponding to a pre-named field to the shippable software module for

insertion into the field in the electronic form when the intended-practice condition of the field is consistent with the use-preference condition of the pre-named field.

3. The method as recited in claim 1 wherein receiving a request from a remote computer further comprises receiving a user identifier associated with the particular user and a form identifier associated with the electronic form, and wherein the remote computer issues the request when the electronic form is downloaded by the particular user.

4. The method as recited in claim 1 further comprising declining to attach a data string from the raw data file corresponding to a pre-named field to the shippable software module for insertion into the field in the electronic form when the intended-practice condition of the field is inconsistent with the use-preference condition of the pre-named field.

5. The method as recited in claim 4 wherein the intended-practice condition and the use-preference condition are collections of type-of-use descriptors.

6. The method as recited in claim 1 wherein dynamically constructing a shippable software module suitable for execution on the remote computer for inserting data strings into an electronic form utilizing the form mapping and the raw data file further comprises examining a plurality of negotiation objects for one of an acceptance and decline, each negotiation object resulting from comparing data relating to the field in the electronic form to data relating to the pre-named field.

7. The method as recited in claim 6 further comprising constructing a JavaScript module from each negotiation object resulting in an acceptance.

8. The method as recited in claim 1 wherein the shippable software module is a JavaScript program.

9. A server for enabling the automatic insertion of data strings into an electronic form having a plurality of fields displayed on a remote computer capable of communicating with the server, the server comprising:

a first memory area for storing a plurality of raw data profiles, each raw data profile corresponding to an associated registered user;

a second memory area for storing a plurality of form mappings, each form mapping corresponding to an associated registered form;

a comparison module for comparing user-preference data determined by an associated registered user contained in the plurality of raw data profiles with practice-preference data determined by a form-originating server contained in the plurality of form mappings; and

a software module constructor for dynamically constructing and transmitting a shippable program suitable for execution on a remote computer to insert data strings into an electronic form on the remote computer.

10. The server as recited in claim 9 wherein each raw data profile includes a plurality of standard field names, each standard field name having a corresponding data string and a use-preference data item as determined by the associated registered user.

11. The server as recited in claim 10 wherein each form mapping includes a plurality of non-standard field names from the electronic form, each non-standard field name being mapped to a standard field name and having a practice-preference data item as determined by a form provider.

12. The server as recited in claim 9 wherein the comparison module contains a plurality of conditions from which the practice-preference data and the use-preference data are defined.

13. The server as recited in claim 9 further comprising a negotiation history module containing a plurality of negotiation modules, each negotiation module containing an offer component and one of an acceptance component and a decline component.

14. The server as recited in claim 13 wherein the acceptance component further comprises a data string corresponding to at least one of the plurality of fields in the electronic form.

15. The server as recited in claim 9 wherein the shippable program is a JavaScript program containing instructions regarding fields in which to insert each data string in the electronic form on the remote computer.

16. The server as recited in claim 15 wherein the shippable program further comprises the plurality of data strings to be inserted in the electronic form.

17. The server as recited in claim 9 further comprising a registered user index for locating a particular raw data profile given a user identifier corresponding to a registered user.

18. The server as recited in claim 17 wherein the user identifier is a cookie.

19. The server as recited in claim 9 further comprising a form mapping index for locating a particular form mapping given an electronic form identifier.

20. The server as recited in claim 19 wherein the electronic form identifier is a Uniform Resource Locator.

21. A method for constructing a shippable software module on a personal information server suitable for execution on a remote computer for inserting data strings into an electronic form, the method comprising:

receiving a request from a remote computer for a shippable software module suitable for execution on the remote computer for inserting data strings into the electronic form;

retrieving a form mapping containing a plurality of associations between fields in the electronic form and pre-named fields on the personal information server, the mapping being associated with the electronic form;

retrieving a raw data file containing data strings from a plurality of raw data files, each data string corresponding to a pre-named field and the raw data file being associated with a particular user;

dynamically constructing a shippable software module suitable for execution on the remote computer for inserting data strings into an electronic form utilizing the form mapping and the raw data file;

transmitting the shippable software module to the remote computer;

registering one or more electronic forms prior to retrieving a form mapping, wherein registering one or more electronic forms comprises obtaining an electronic form so that a form mapping for a plurality of associations between fields in the electronic form and pre-named fields on the personal information server can be generated;

obtaining intended-practice conditions for each of a plurality of fields in the electronic form; and

embedding within the electronic form an executable linking module capable of sending a request from the remote computer to the personal information module for a shippable software module suitable for execution on the remote computer for inserting data strings into the electronic form.

22. A method for constructing a shippable software module on a personal information server suitable for execution

on a remote computer for inserting data strings into an electronic form, the method comprising:

receiving a request from a remote computer for a shippable software module suitable for execution on the remote computer for inserting data strings into the electronic form;

retrieving a form mapping containing a plurality of associations between fields in the electronic form and pre-named fields on the personal information server, the mapping being associated with the electronic form;

retrieving a raw data file containing data strings from a plurality of raw data files, each data string corresponding to a pre-named field and the raw data file being associated with a particular user;

dynamically constructing a shippable software module suitable for execution on the remote computer for inserting data strings into an electronic form utilizing the form mapping and the raw data file; and

transmitting the shippable software module to the remote computer;

wherein the particular user is previously registered with the personal information server, and wherein registering the particular user with the personal information server comprises:

providing the particular user's raw data corresponding to pre-named fields on the personal information server; and

providing use-preference conditions for the particular user's raw data corresponding to each of the plurality of pre-named fields.

* * * * *

US005794259A

United States Patent [19]

Kikinis

[11] Patent Number: 5,794,259

[45] Date of Patent: Aug. 11, 1998

[54] **APPARATUS AND METHODS TO ENHANCE WEB BROWSING ON THE INTERNET**

[75] Inventor: **Dan Kikinis**, Saratoga, Calif.

[73] Assignee: **Lextron Systems, Inc.** Saratoga, Calif.

[21] Appl. No.: **688,022**

[22] Filed: **Jul. 25, 1996**

[51] **Int. Cl.**[6] **G06F 7/06**

[52] **U.S. Cl.** **707/507**; 707/513; 395/200.48; 705/26; 345/352

[58] **Field of Search** 395/200.48; 707/505–507, 707/501; 345/352; 705/26, 27

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,325,478	6/1994	Shelton et al.	707/507
5,404,294	4/1995	Karnik	707/507
5,450,537	9/1995	Hirai et al.	707/507
5,523,942	6/1996	Tyler et al.	707/507
5,555,325	9/1996	Burger	382/309
5,625,465	4/1997	Lech et al.	358/448
5,640,577	6/1997	Scharmer	395/768
5,666,502	9/1997	Capps	345/352

OTHER PUBLICATIONS

CGI Programming on the WorldWide Web Gundavaram pp. 407–411. Mar. 1996.

Special Edition Using Netscape™ 2 Brown et al. pp. 263–285 & 786–814. 1995.

Primary Examiner—Larry D. Donaghue

Attorney, Agent, or Firm—Donald R. Boys

[57] **ABSTRACT**

A system for filling fields in Internet forms follows executable control code to associate stored fill entities with field names, and to place the stored fill entities into fields in the Internet form. In one embodiment association is automatic to the extent that names of fill entities match field names in the form. In another embodiment a display list is provided superimposed on the form, the display having selectable stored entity names. In this embodiment of the invention entity names may be selected from the superimposed list and caused to fill selected fields sin the form. In some embodiments both features are provided. In yet another embodiment a WEB browser is adapted to download database entities from a remote server through an Internet connection directly to a memory que without immediate display. The stored entities are separately selectable from the memory que for display and processing independent of operation of the WEB browser or the Internet connection.

5 Claims, 4 Drawing Sheets

Beginning of document ...

line n < P > < FONT SIZE = + 1 > Your E-mail address: < /FONT >

line n + 1 < BR > INPUT NAME = "E-MAIL" TYPE = "TEXT" ROWS = 1 SIZE = 40 >

line n + 2 < P > < FONT SIZE = + 1 > Your name: < /FONT >

line n + 3 < BR > < INPUT NAME = "Name" TYPE = "TEXT" ROWS = 1 SIZE = 40 >

line n + 4 < P > < FONT SIZE = + 1 > Company Name < /FONT > (if applicable):

line n + 5 < BR >< INPUT NAME = "Com name" TYPE = "TEXT" ROWS = 1

SIZE = 40 >

.....end of document

Beginning of document ...

line n < P > < FONT SIZE = + 1 > Your E-mail address: < /FONT >

line n + 1 < BR > INPUT NAME = "E-MAIL" TYPE = "TEXT" ROWS = 1 SIZE = 40 >

line n + 2 < P > < FONT SIZE = + 1 > Your name: < /FONT >

line n + 3 < BR > < INPUT NAME = "Name" TYPE = "TEXT" ROWS = 1 SIZE = 40 >

line n + 4 < P > < FONT SIZE = + 1 > Company Name < /FONT > (if applicable):

line n + 5 < BR >< INPUT NAME = "Com name" TYPE = "TEXT" ROWS = 1

SIZE = 40 >

.....end of document

Fig. 1

10/02/2003, EAST Version: 1.04.0000

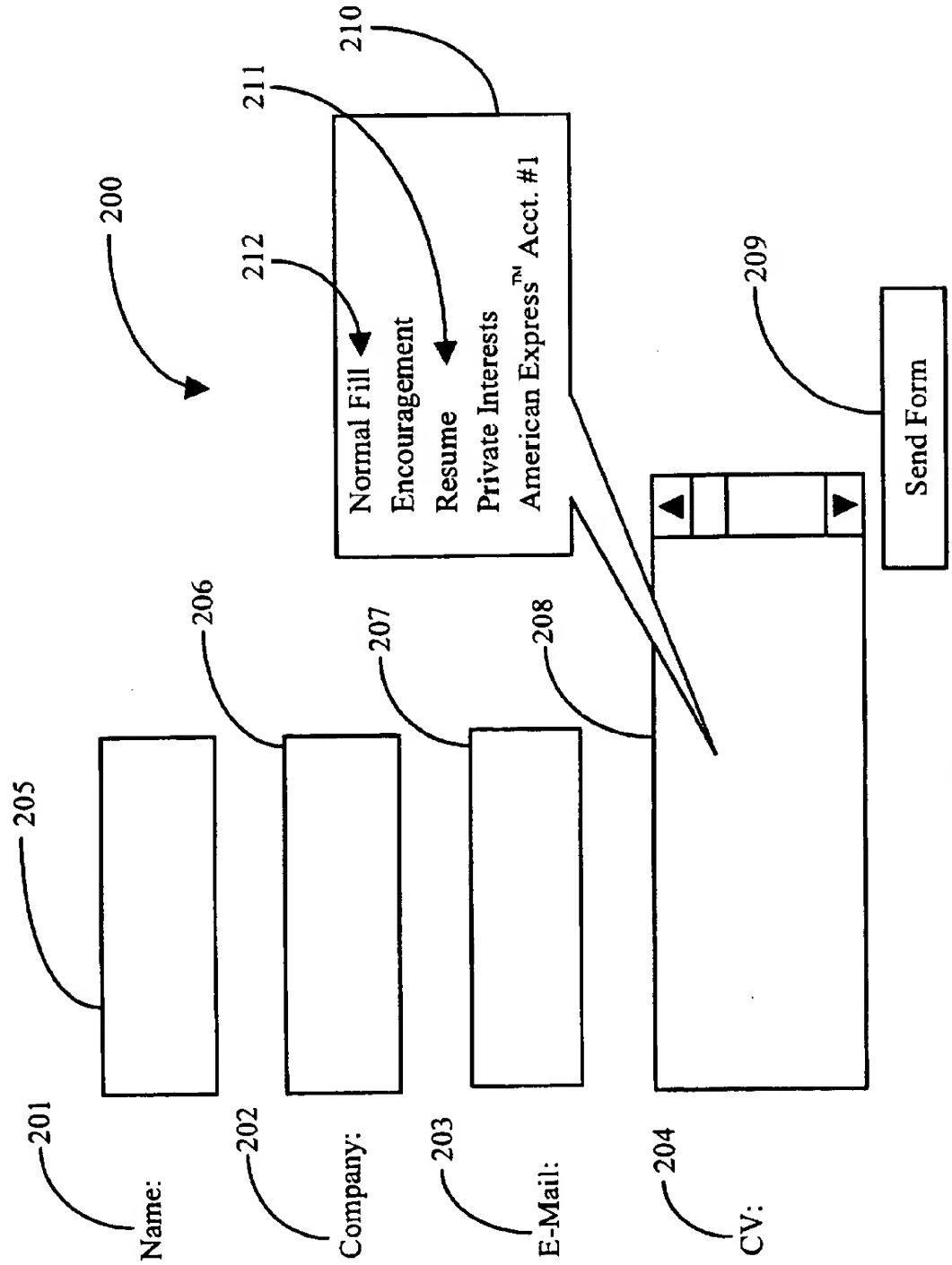

200
201
Name:
202
Company:
203
E-Mail:
204
CV:
205
206
207
208
209
Send Form
210
211
212
Normal Fill
Encouragement
Resume
Private Interests
American Express™ Acct. #1

Fig. 2

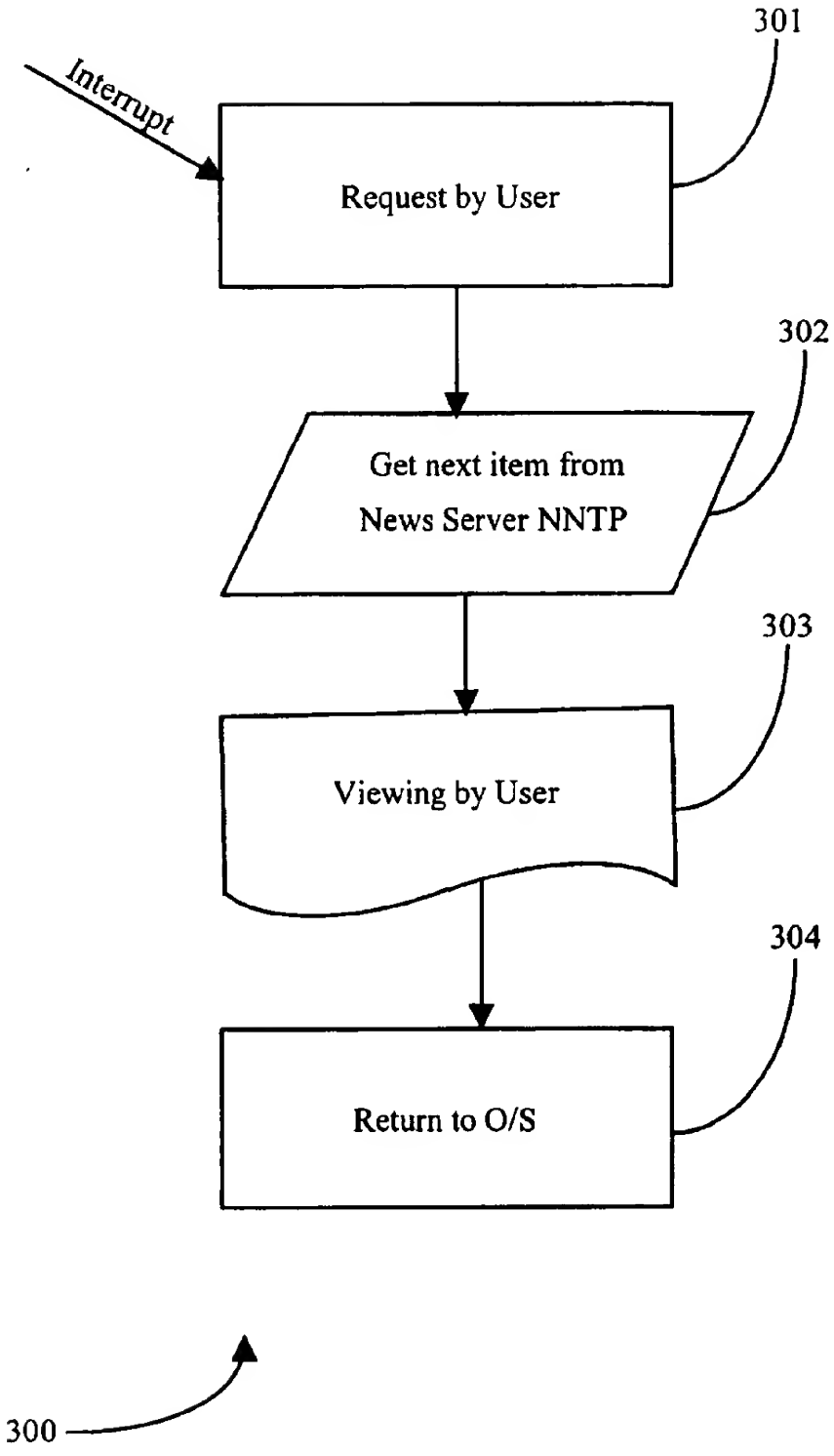

301
Interrupt
Request by User
302
Get next item from
News Server NNTP
303
Viewing by User
304
Return to O/S
300

Fig 3A

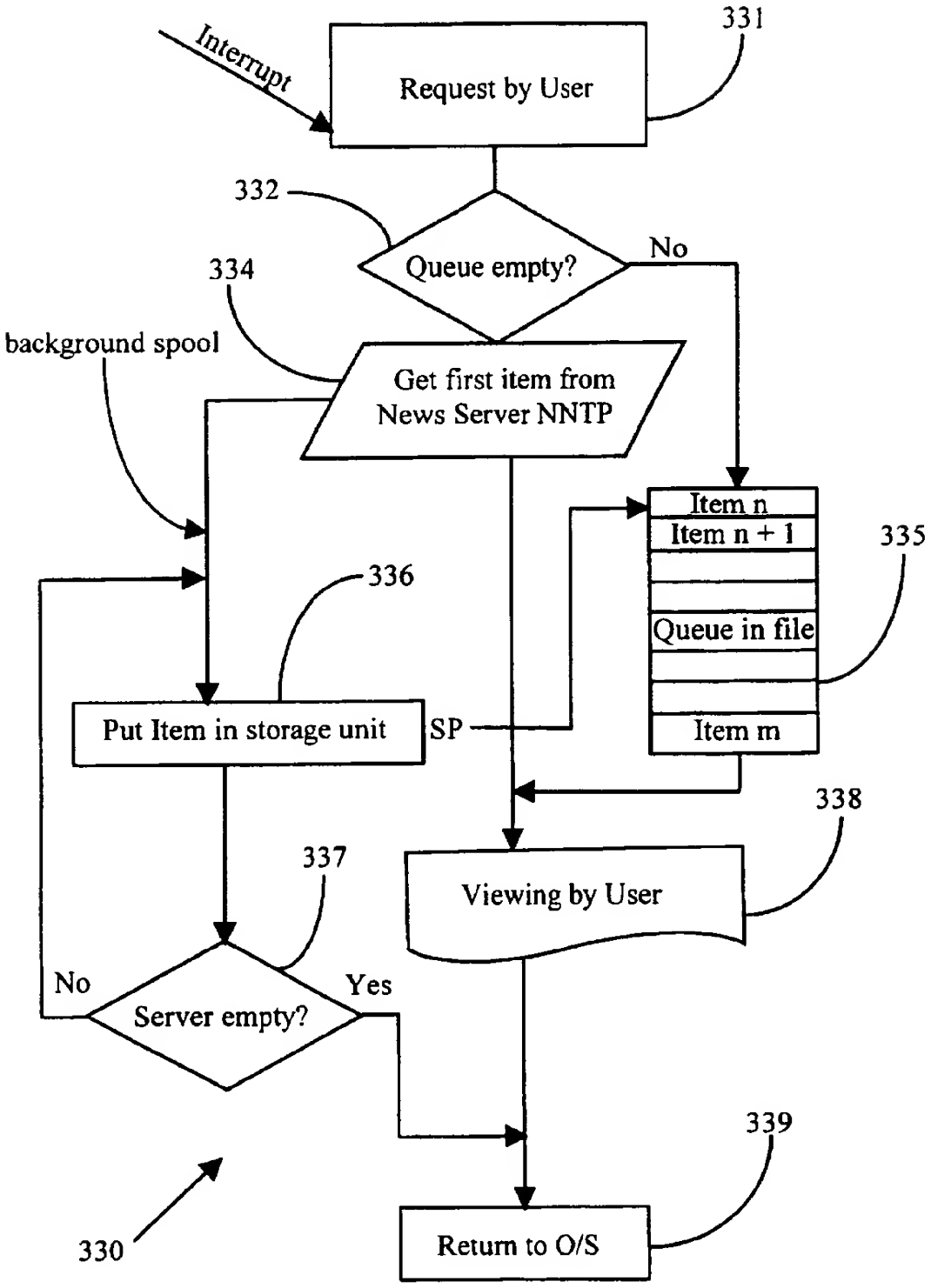

Interrupt
Request by User
331
332
Queue empty?
No
334
background spool
Get first item from
News Server NNTP
Item n
Item n + 1
335
336
Queue in file
Put Item in storage unit
SP
Item m
338
337
Viewing by User
No
Server empty?
Yes
339
330
Return to O/S

Fig. 3B

APPARATUS AND METHODS TO ENHANCE WEB BROWSING ON THE INTERNET

FIELD OF THE INVENTION

The present invention is in the field of Internet World Wide WEB (WWW) browser technology, and pertains more particularly to tools for filling in forms and downloading information from the WWW.

BACKGROUND OF THE INVENTION

Part of the global network termed the Internet, called the World Wide Web (WWW or WEB) has a lot of information and services available to users. Often when browsing the WEB, one encounters forms to fill out, mostly for buying or subscribing to services or products. These forms are often long, asking for lots of information, and tedious to fill out. One reason there are so many forms is that very often, services that are provided as free are not free at all. Instead, one trades information for give-away software and or services.

The goal of marketing schemes on the Internet, as is true elsewhere, is to collect as much information about potential or actual customers as is possible or practical. Such information can lead to future sales, and can also be sold to other organizations. Hence during long browse sessions on the WEB, one may have to fill in several forms, mostly asking for the same or similar information over and over. A person actively using the WEB may waste quite some time in filling out forms.

In another aspect of Surfing the Net (using the Internet) users access news groups, which are very good places to trade information with mostly other users about common interests. A myriad of different news groups have developed, and any one person can easily be interested in 5–10 or even many more news groups. In accessing news groups there is a problem in the way one has to download information. Typically a file is first downloaded containing headers only. Then, as the user selects items, each item is specifically downloaded. Very often it can take 5–10 seconds or more, before the next item is downloaded. On very active news groups, that have more than 200 postings a day, that accumulated time can be considerable, limiting the usefulness of the Internet.

What is needed in the first instance described above, that of filling in forms on the Internet, is a system that allows a user to link specific pre-stored data, usually data unique to the user, with fields in forms encountered on the Internet, such that a pipeline is established for quickly and efficiently filling fields in forms.

In the second instance described above, that of accumulated delay in downloading information from the WWW, such as from newsgroups, what is needed is a caching system whereby information from newsgroups may be prefetched and stored locally, that is, at the user's computer, where it may be accessed off-line.

SUMMARY OF THE INVENTION

In a preferred embodiment of the invention a system for filling fields having field names in forms encountered on the Internet is provided, comprising: a central processing unit (CPU); a display operable by the CPU; stored fill entities associated with tags accessible by the CPU; and control code executable by the CPU. The CPU, executing the control code, compares the field names in the forms with the tags of the stored fill entities, and places fill entities in fields wherein the tags match the field names.

In a further embodiment of the invention the system comprises a display bubble displayable by the CPU executing the control code, the display bubble positionable by a user on the display to select fields to be filled, the display bubble displaying multiple tags for stored fill entities, the tags selectable by the user to cause the stored entities associated with the tags to fill the fields selected by the user.

The apparatus thus arranged makes possible a method for filling out forms on the Internet comprising steps of (a) selecting a field in an Internet form; (b) invoking a display of tags associated with multiple stored fill entities; (c) selecting one of the tags in the display of tags, causing the fill entity associated with the tag to be entered in the selected field in the form; and (d) repeating steps (a) through (c) until all of the fields in the form are filled. Another method according to the invention does not use the display of tags of fill entities, but instead relies on an automatic association of entity tags with field names, with field fill being accomplished to the greatest extent possible transparent to the user. The user may afterward check and correct the filled fields.

In another aspect of the invention a system for downloading and displaying data through an Internet connection from a remote server on the Internet is provided, comprising a computer having a memory and a display screen; and a WEB browser application adapted for displaying names of database entities on the remote server and having a facility for a user to select names of database entities to be downloaded. Database entities selected to be downloaded are downloaded to the memory rather than immediately displayed, and display of downloaded entities is separately selected from the memory, such that entities may be viewed from the memory while other entities are being downloaded, and may continue to be selected and viewed after the Internet connection is closed.

An associated method for downloading data entities from a server remote from a computer station comprises steps of (a) logging on to a remote server through an Internet connection; (b) selecting database entities to be downloaded and displayed; (c) downloading the database entities directly to a memory file rather than immediately displaying said database entities; and (d) separately selecting the database entities from the memory file for display independently of the downloading process and of the Internet connection.

Using these unique WEB browser enhancements provides speed and facility in downloading data and in filling out forms that has heretofore not been available to Internet users, and saves valuable time, hence cost in using the Internet.

BRIEF DESCRIPTION OF THE DRAWING FIGURES

FIG. 1 is an example of HTML script associated with an on-line form on the WEB.

FIG. 2 is an illustration of a browsing tool for filling in Internet forms according to an embodiment of the present invention.

FIG. 3A is a flow diagram depicting the conventional process of accessing data from the Internet.

FIG. 3B is a flow diagram depicting a method of accessing information from the Internet according to an embodiment of the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Since a computer in conjunction with browsing the Internet is mainly a productivity or entertainment tool, a user

wants to enforced wait times be as short as possible. Very often, if there are too many waits of too long duration, or if forms to be filled in are too cumbersome, the user will simply go away, and do something else, much like a customer not served quickly in a restaurant may get up and leave. What is needed are ways to reduce idle times in browsing as much as possible, but in a way that is entirely backwards compatible. The existing investment into the infrastructure of the Internet has reached a point where it would not be possible to make radical changes.

As described above in the Background section, users encounter many forms to be filled out in return for access to information or executable software that may be downloaded from servers on the Internet. Filling out such forms can be time consuming and boring.

A unique solution to the difficulty of form filling, according to an embodiment of the present invention, takes advantage of the fact that programmers typically use descriptive names in preparing WEB pages. FIG. 1 shows a small section of typical HTML3.0 code (Hyper Text Markup Language revision 3.0) as may be associated with a part of a WEB page. In line n+1 the field name n"Email" is clearly related to the label in line n "Your Email address", even though there is no strict convention in programming to provide such relationships. The reason is that programmers mostly use descriptive names in order to make code more readable. Following such an approach, it is quite possible that once the method described in this disclosure is available, an official appendix to future revisions of HTML may have a standard convention for the most used field names in forms.

FIG. 2 is an illustration of an interactive tool for use in filling in form fields in forms encountered on the Internet. The tool according to an embodiment of the present invention allows a user to quickly link pre-stored information of the sort most usually required by forms to fields in forms, and to transfer such information to the form fields.

FIG. 2 shows a portion of a form as might be encountered on the Internet, including fields **205** labeled "Name" **201**, **206** labeled "Company" **202**, **207** labeled "E-Mail" **203**, and **208** labeled "CV" **204**. There is, in addition, an interactive button **209** labeled "Send Form" enabling a user to transmit a filled-in form.

As is well known in the art, the conventional process of filling in such a form is to move an on-screen cursor to a field, and to type in alphanumeric characters in the field. In an embodiment of the present invention, control code is provided to automatically fill in such forms when user activated. The control code may be a terminate-and-stay-resident (TSR) program, for example, or a plug-in module to a WEB browser application. In a preferred embodiment the control code of the invention is a plug-in to a WEB browser. It will be apparent to those with skill in the art how such a plug-in module is associated with a browser, without the necessity of providing such conventional details in this specification.

In a preferred embodiment of the present invention customized for a particular user, information, such as name, address, home phone number, business phone number, facsimile number, E-Mail address, company, and so on, is stored and accessible by the control code by association with a name tag. When a user encounters a form on the Internet, and wishes to fill in the form, he/she hits a "hot key", or key combination, which invokes the control code of the invention. The code executing matches field names in the form with tags to the prestored information about the user, and fills all of the fields for which a match is made. In some cases this may be all of the fields in the form. In other cases, some matches will not be found, or an inappropriate match will be made, and incorrect information will be entered in the form.

At this point the user need only peruse the filled-in form for accuracy. If incorrect information is in a field, it may be corrected. In a preferred embodiment another Hot Key or key combination, or key and mouse button combination causes the control code executing to display a bubble **210** having a selection list **212** of tags for prestored information. FIG. 2 shows how the bubble **210** could look on the screen, after being activated by holding a key and clicking a mouse button. The user can move a highlight bar **211** up and down, and select an item to be pasted into the field where tip (**208**) is pointing. It will be apparent to those with skill in the art that there are a variety of ways the bubble feature may be activated. For example, the code could be provided so one may move the conventional screen cursor to a field and provide the activating signal by a hot key, displaying the bubble. Movement of the highlight bar could then be by further cursor movement, as in drop-down menus, or by arrow keys. Any one of a variety of mechanisms might be incorporated for selection of a highlighted item in the list, which then is inserted into the field to which the bubble points.

In the example shown here the user is about to paste the content of the file tagged "resume" from a list **212** into the field **208** labeled CV (Curriculum Vitae) **204**, with a single click of a button In an alternative embodiment the bubble may be invoked at the first use of the control code, and used with the "Normal Fill" selection to fill fields one at a time. Because several users may share the same system, the data is preferably stored in an encrypted manner, allowing a user to log-in as well as password protecting such access. By activating such a user profile, all other parameters may also be personalized, such as e-mail name, mail server, news server etc.

In another aspect of the present invention provision is made to enhance operation of downloading information from databases such as news groups on the Internet. FIG. 3A illustrates conventional operation of a process **300** of downloading data entities from an On-line news group at a remote server (remote from the users station). When a user logs on to a news group, a menu of indexed entities is displayed for selection by the user. The user selects an entity at step **301**, and that entity is downloaded (Step **302**) at whatever data rate has been established between the remote server and the user's platform. The entity is displayed on the user's screen and viewed by the user (Step **303**). The user has a number of options, such as sending the displayed entity to storage or to a printer, or both, but nothing more happens until the user finishes with the first entity (**304**), and them selects a new entity to download.

Procedure **330** in FIG. 3B shows an enhanced procedure according to a preferred embodiment of the present invention. After a user request **331** the status of a queue or buffer **335** is checked. If the buffer is empty, a unique background process is started, mainly consisting of steps **336** and **337**, downloading selected newsgroup entities into a queue **335**, which may be a file on disk, but can also be in other suitable storage or memory. Meanwhile, the user can view at leisure out of queue **335** , as long as items are available. The viewing process can continue without an operating connection to the server, which means the user can view the items in **335** "off-line".

It will be apparent to those with skill in the art that there are many variations to the embodiments described above

which may be made without departing from the spirit and scope of the invention. Several such alternatives have already been described, such as the several ways in which a fill-in bubble may be invoked. It is well-known, too, that there are many personal preferences in programming, and that like results may be obtained by alternative coding schemes. There are likewise many forms that the fill-in bubble of the invention might take. There are similarly many alternative implementations that may be made for the enhanced downloading technique described above.

What is claimed is:

1. A system for providing data for fields having coded field names in forms downloaded as code from a server on the Internet, comprising:

a central processing unit (CPU);

a display operable by the CPU;

stored fill entities associated with tags accessible by the CPU; and

control code executable by the CPU;

wherein the CPU, executing the control code, compares the coded field names in the downloaded code with the tags associated with the stored fill entities, retrieves fill entities when a match is made, and associates retrieved fill entities with fields wherein the tags match the field names, preparatory to transmission to the server on the Internet.

2. The system of claim 1 further comprising a display bubble displayable by the CPU executing the control code, the display bubble displaying multiple tags associated with stored fill entities, wherein the user may associate any field with the display bubble, any one tag in the display bubble then being selectable by the user to cause the stored fill entity associated with the selected tag to fill the field associated with the bubble.

3. A method for providing data for fields having coded field names in forms downloaded as code from a server on the Internet and displayed on a display monitor, comprising steps of:

(a) invoking a display of tags associated with multiple stored fill entities;

(b) selecting a field in the displayed Internet form, thereby associating the field with the display of tags;

(c) selecting one of the tags in the display of tags, causing the stored fill entity associated with the selected tag to be entered in the selected field in the form; and

(d) repeating steps (a) through (c) to fill additional fields in the form.

4. A method incorporating a computer having a display screen, the method for providing data for forms having fields with coded field names downloaded as code from a server on the Internet comprising steps of:

(a) associating fill entities with tags;

(b) storing the fill entities associated with the tags in a memory of the computer;

(c) downloading a form as code from a server on the Internet through a browser using the computer;

(d) associating the tags of the stored fill entities with coded field names in the code of the downloaded form; and

(e) causing the stored fill entities with tags matching coded field names in the code of the downloaded form to be associated with the coded fields to which the tags match.

5. The system of claim 2 wherein the association of the display bubble with a field on the form is accomplished by positioning the display bubble to contact the field on the displayed form.

* * * * *